Machine Conceptualization Of Natural Language Text

A thesis submitted to the
Division of Research and Advanced Studies
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

MASTERS OF SCIENCE

in the Department of Electrical and Computer Engineering and Computer Science
of the College of Engineering

1995

by

Joe Lohner

B.B.A., College of Business, University of Cincinnati

Committee Chair: Dr. Larry Mazlack
Committee Member: Dr. Raj Bhatnagar
Committee Member: Dr. John Schlipf

Abstract

The idea that a computer can understand natural language text is examined in relation to its ability to conceptualize it. Conceptualization involves many cognitive processes such as the ability to recognize related things, create a taxonomy, and determine a syntactic object's relative importance. To do this, it is useful to identify: categories, best examples of an object within a category, and underlying building blocks. This thesis measures to what extent a computer can conceptualize natural language text using an unsupervised semantic net formation technique. This was done by using several steps. Algorithms were developed to form a semantic net from natural language text input. A cognitive method was created to categorize objects within the semantic net. Five cognitive procedures were developed to rank objects by importance. Four cognitive methods were developed to rank objects by how well they would make best examples of a specific category. A cognitive routine was created to select basic-level objects. Statistical measures were defined to calibrate the machine's ability to execute the aforementioned cognitive processes in comparison to a class of college students. The machine's cognitive methods were then executed using natural language text for input to create conceptualization output. Next, the machine's conceptualization output was calibrated in comparison to a college student conceptualization survey. Finally, the results of these comparisons and the machine's cognitive output were then described and analyzed to determine to what extent a machine can conceptualize natural language text.

Acknowledgments

I would like to thank foremost, my parents for their never ending support in my venture in getting a master's degree. Next, I would like to thank Vera, because she's the one.

Now for the technical side of things, I would like to thank Dr. Larry Mazlack for his time and advice in the development of my thesis. I also would like to thank Dr. Bhatnagar and Dr. Schlipf for their advice. Many thanks go to Dr. Steve Howe from the University of Cincinnati, College of Arts and Sciences, Psychology Department and Dr. Jim Deddens from the University of Cincinnati, College of Arts and Sciences, Mathematics Department for their help in developing statistical methods used in this thesis.

One more big thanks goes out to the University of Cincinnati's College of Engineering for their financial support through this endeavor. I would especially like to recognize the folks in the engineering's OCC.

Table of Contents

Figures

Tables

Examples

Algorithms

1. Introduction

Since the dawn of computing, it has been a goal for a computer to understand natural language. Natural language is the everyday, informal language that people use to communicate with each other.

For a computer to understand a natural language, it has to be able to conceptualize. Conceptualization involves many cognitive processes such as the ability to recognize related things, create a taxonomy, and determine a syntactic object's relative importance. To do this, it is useful to identify: categories, *best examples* of an object within a category, and underlying building blocks.

This thesis measures to what extent a computer can conceptualize natural language text using an unsupervised semantic net formation technique.

Several steps were required to perform this computer conceptualization assessment. First, a computer program was written that could perform several of the aforementioned cognitive processes upon natural language text. The program runs unsupervised and creates a semantic net to formulate and identify concepts within. Next, the program produces several conceptual results after receiving natural language text. Finally, the computer's performance was evaluated by taking its conceptualization results and comparing them to human conceptualization results.

Evaluating human and computer cognitive processes on equal footing was straightforward. The computer and the human subjects were given third grade reading material in which they both had the vocabularies to understand what they were reading. The humans received a printed article to read while the computer was given the same article in a text file. When the students were finished reading the article they were asked to answer several questions. The computer on the other hand was prompted for several printed results after reading the same article. Both student answers and computer results were converted into numerical data and compared with one another to perform a conceptualization evaluation.

Going from a text file that contains natural language (NL) text to formulating and identifying concepts is a major transformation and requires three major processing steps. The first step, Natural Language Text Processing, involves text parsing, word sense identification, book structure formation, phrase parsing, and pronoun resolution. The second step, Semantic Net Construction, of the transformation process takes the pronoun resolved phrase structure and constructs and manipulates a semantic net. Finally, the third step, Concept Identification, reads in the semantic net and identifies entity categorization, entity importance, *best examples* and *basic level* entities. All three steps use a built in dictionary and an attribute table.

Natural Language Text Processing involves the implementation of five minor steps executed in the following order. The first step, text parsing, reads in the text file or user input and parses each sentence into separate words and punctuation. The second step, word sense parsing, assigns the proper dictionary definition to each word node. The third step, book structure construction, inserts the word nodes into their appropriate sentence and paragraph structures. The fourth step, phrase parsing, transforms the book structure into many phrase structures, where each phrase consists of a subject, action, and object (if it exists). The fifth and final step, pronoun resolution, assigns noun definitions to pronoun word nodes wherever possible.

Semantic Net Construction entails two steps, Semantic Net Formation and Semantic Net Modification. The first step, Semantic Net Formation, constructs or "adds to" (if the net already exists) the semantic net using the pronoun resolved phrase structure. This step converts phrases into semantic relationships and adds them into their appropriate places within the net. The second step, Semantic Net Modification, cleans up after the Semantic Net Formation by determining an entity's primary parent,

generalizing entity attributes up the net, making sub-category names more descriptive, and combining duplicate categories.

Concept Identification encompasses four cognitive processes: entity categorization, entity importance, *best example* identification and *basic level* selection. The entity categorization process determines which category an entity belongs under. The entity importance process ranks a subset of entities by importance. *Best Example* entity process ranks a subset of entities as to which is the *best example* of a particular category. *Basic Level* entity process chooses *basic level* categories or entities. The computer output of all four cognitive processes is then compared to human results for evaluation.

The evaluations of all four cognitive processes are summarized and elaborated upon to determine to what extent a machine can conceptualize natural language text. The results came from converting the human and computer answers into a numerical format. High level statistics were then calculated using the numerical results to determine a measure of similarity between the computer and human results. If the similarity between the tested humans and the computer were high then it can be said that the computer had some conceptualization ability, otherwise it did not.

1.1. Thesis Mini-Guide

- Introduction - Chapter 1.
- Hypothesis - Chapter 2.
- Dictionary Structure - Chapters 3 and 4.
- Natural Language Input - Chapter 5.
- Parsing Processes - Chapters 6 through 9.
- Pronoun Resolution - Chapter 10.
- Semantic Net Construction - Chapters 11 and 12.
- Conceptualization - Chapters 13 through 16.
- Conclusion - Chapter 17.

2. Hypothesis

2.1. Research Question

To what extent can a machine conceptualize natural language text using an unsupervised semantic net formation technique.

Hypothesis

A computer can be programmed to conceptualize natural language text similar to the average student. The natural language used in this experiment is elementary grade science books. Conceptualization is defined as the ability to create a taxonomy, determine a syntactic object's relative importance, identify *best examples* of an entity within a category, and select *basic-level* objects.

2.2. Definitions

- ***Conceptualization*** - Conceptualization is the ability to:
  - Create a taxonomy from natural language text.
  - Determine what is the most important concept from a subset of concepts.
  - Determine what is the *best example* of a concept from a subset of concepts.
  - Capture concept attributes.
  - Determine *basic level* entities.

- ***Best Example*** - The *Best Example* is the object that gives the best general description for the rest of the objects in the category. It also can be the object used as a stereotypical example for the rest of the objects in the category.

  For example:
  Given a group of houses:
  cape cod house, White House, single story ranch, Fred's house,
  school house, and a two story red brick house
  The *Best Example* of a house could be a single story ranch.

- ***Most Important*** - The *Most Important* object is the object that:
  - exemplifies the best traits
  - has the most known about it
  - does the most

  For example:
  Given a group of houses:
  cape cod house, White House, single story ranch, Fred's house,
  school house, and a two story red brick house
  The *Most Important* house could be the White House.

- ***Basic-Level*** - *Basic-Level* objects: are the easiest and the first to be learned by children; they encapsulate our gestalt perception and bodily movement in their descriptions; they carry the most information; they are the most differentiated from each other; their subordinate members have: the most attributes in common, similar shapes, and motor movements that are similar to one another; are based on large clusters of correlated attributes that overlap very little from category to category. In other words, they are underlying building blocks or can be referred to as semantic primitives. [Lackoff, 1987, pp. 49, 267-268; Gelman, 1988, p. 68; Rosch, Mervis, 1975, p. 602; Rosch, Mervis, 1976, p. 382; Mervis, 1987, p. 202; Berlin, Breedlove, Raven, 1974, pp. 25-45]

  Some examples:

  Nouns - chair, table, house, book, lamp, coat, apple, etc.
  Actions - running, walking, eating, drinking, etc.
  Attributes - tall, short, hard, soft, heavy, hot, cold, etc.
  Colors - black, white, red, green, blue, yellow, etc.
  [Lackoff, 1987, pp. 270-271; Cruse, 1975, p. 153]

- ***Relationship*** - A relationship consists of two entities connected by a relation that are located within a semantic net. One of the entities is referred to as the subject entity that performs an action (relation) upon the object entity. The object entity is the recipient of the action (relation). In a relationship, there does not always need to be an object.

2.3. Measurables

Identification of concepts as to:

- Category membership.
- Determining the *best example* of a concept class.
- Determining the most important concept of a concept class.
- Determining which concepts are *basic level* entities.

2.4. Literary Search

A Literary search was completed before the start of this thesis. The topics that were researched with their corresponding appendices are:

- *Conceptualization Methodologies* - Appendix K
- *Knowledge Representation* - Appendix L
- *Conceptualization Problems* - Appendix M.

The methodologies, representations, and problems that were discovered were summarized and put into their appropriate tables in the appendices.

3. Dictionary

3.1. Scope

- The dictionary is used for:
  - Determining word sense
  - Resolving pronouns
  - Calculating entity importance
  - Indicating which entity is the *best example* of a subset of entities
  - Inferring relationships up the semantic net hierarchy.
  - Determining if compound words should be compressed into one word.
- The dictionary uses inheritance from the definition structure and values from the attribute table to establish the meaning of words.
- The dictionary is pre-defined in terms that are both common and primitive.
- The dictionary defines every character string that is contained in the natural language texts used for this experiment.

3.2. Structure

- An example of the dictionary structure is displayed below.

Figure 3.1 - Illustration of the dictionary in memory

- The dictionary is made up of two interconnected structures, the lexicon and the definition structure.

3.2.1. Lexicon

3.2.1.1. Purpose

The lexicon serves several purposes:

- It serves as a definition structure index, locating words in the definition structure with all their different senses.
- It also translates definition structure nodes into natural language text.
- It is referenced by both the definition structure and the attribute table.

3.2.1.2. Structure

- The lexicon consists of two data structures, the *lexicon root* and the *lexicon entry*.
- The *lexicon root* structure consists of an array of 4226 *lexicon entry* pointers. The 4226 *lexicon entries* represent every two character combination of a-z, A-Z, 0-9, space, apostrophe, and dash.

- The *lexicon entry* structure contains the following data:
  - a pointer to a character string
  - an array of *lexicon entry* pointers
  - a *lexicon entry* pointer array counter
  - an array of definition pointers
  - a definition pointer array counter
- The *lexicon entries* are divided into two levels. The first level containes 4226 *lexicon entries* referenced by the *lexicon root* that represent every possible two character (characters mentioned above) combination. Each of these entries then reference an unique array of *lexicon entries* (second level) that represent all the words that begin with same two characters as their first level counterpart.

  For instance, given the words "than", "thorn", and "throw"; "than" would be stored at the first level lexicon entry representing the "th" index since it comes before "thorn" and "throw"; "thorn" and throw" would then be stored in the second level lexicon entries referenced by the first level lexicon entry "than".
- The second level lexicon entries are sorted in alphabetical order.
- The second level lexicon entry's array of lexicon entry pointers are all set to null and their lexicon entry array counter is always equal to zero. The first level lexicon entry's array of lexicon entry pointers are either pointing to other lexicon entries or are null if no other lexicon entries are referenced.
- The lexicon entry's character string pointer references the natural language word represented by that entry.
- Each lexicon entry contains an array of pointers that reference the different meanings a word might have. The pointers address nodes in the definition structure.
- An example of several lexicon entries is shown below.

Figure 3.2 - Sample Lexicon Entries

3.2.2. Definition Structure

3.2.2.1. Purpose

- The definition structure defines many different meanings a natural language word can have. It is also referenced by the *word sense* rule base and the attribute table to further define meanings.

3.2.2.2. Structure

- The definition structure is an unary hierarchical tree.
- The arcs connecting the nodes are bi-directional so the tree can be traversed in ascending (referred to as "isa" arcs) and descending (referred to as "includes" arcs) directions.
- The definition structure is made up of definition nodes which contain the following fields:
  - *Isa Pointer* - A pointer that references the node's parent. Each node has only one parent.
  - *Includes Pointers* - An array of "includes" pointers that references the node's children. Each node can have up to 32,767 children.

- *Lexicon Entry Pointer* - A pointer to the definition node's natural language word located in the lexicon.
- *Attribute Pointer* - A pointer to a list of attributes. The definition node's attribute data structure consists of a pointer to the next attribute, an integer to represent the attribute table index, and an integer to represent the attribute's value. A definition node can have from 0 to ∞ attributes.
- *Reference Word Pointer* - A pointer to a character string that stores the definition node's reference word. It is used to differentiate between the natural language word's multiple meanings. This reference word is constructed by taking the natural language word concatenated with a "#" followed by an integer (incremented by one for each different word sense).

  For example, if there were four senses of the word "run" then there would be four definition nodes to represent each sense of the word "run". Their corresponding reference words would be "run#0", "run#1", "run#2", and "run#3".
- *Definition Structure Level* - An integer that indicates the level of the definition node within the definition hierarchy. The root node is at level zero. The level below that is level one, and so on.
- *Includes Pointer Quantity* - An integer that stores the number of the node's children.
- *Definition Flag* - An integer that indicates if a definition is a base word. If a definition node is a base word, then it is not used as definition itself but rather it has pointers to other versions of itself. These other versions representing different "counts", "genders", and "verb tenses" of the base word. Base words are not used in identifying natural word meanings in the parsing processes.

  For example, take the word "run" in which "run#0" would be the base word. "Run#0" would then have pointers to "run#1", "ran#1", and "running#1". "Run#1", "ran#1", and "running#1" represent the different tenses of the word run.
- *Grammar Type* - An integer that indicates the node's grammar type. This is used in determining a sentence's subject, action, objects of the action and modifiers. Fifteen different grammar identifiers that were used in the experiment are listed in the table below.

Identifier	Grammar Type
0	Noun
1	Verb
2	Adjective
3	Determiner
4	Premodifier
5	Adverb
6	Pronoun
7	Preposition
8	Conjunction
9	Interjection
10	Question Word
11	Negative
12	Auxiliary Verb
13	Syntax
99	Undefined Grammar

Table 3.1 - Grammar Types

- An example of an definition structure node is illustrated below. The corresponding c++ objects are listed in appendix H under "Definition", "LexEntry", and "Attribute".

Figure 3.3 - Sample Definition Structure Node

3.3. Construction

The dictionary is the second data structure to be built right after the attribute table has been loaded in memory. Both lexicon and definition structures are constructed at the same time so that each natural language word is connected with it's appropriate definition structure nodes. The dictionary is created by the following processes.

- First, the dictionary root is created with the word "text" as the lexicon reference and "text#0" for the definition structure's root node. The word "text" is used as the root, since all strings of characters are considered text.
- Second, the dictionary seed file is loaded. This file contains the dictionary references to build all the other words to be defined in the dictionary. The dictionary seed is displayed in appendix D.
- Finally, the rest of the dictionary, using terms defined in the dictionary seed, is loaded in. The dictionary is loaded in several steps, so that different references can be constructed and defined using the dictionary seed.

Dictionary input records have the following format:

[(lexicon word, definition reference word, definitions word's parent reference word)|attribute name, attribute value | attribute name, attribute value |....]

- *lexicon word* - A word stored in the lexicon and is also how the word appears in its natural language form.
- *definition reference word* - The lexicon word concatenated with a "#" followed by a unique integer. Each definition reference word represents the different meanings a natural language word can have. The word is stored within the definition structure.
- *definitions word's parent reference word* - An identifier that indicates where the word is to be stored in the definition structure.
- *attribute name* - The name of an attribute located in the attribute table. A definition can have 0 to n attribute entries.
- *attribute value* - The attribute name's associated value. The values are referenced in the attribute table.

A comma is used to separate the lexicon word, definition reference, and parent reference. This enables word phrases to be defined as a single entity; i.e., "kick the bucket". A "|" is used to separate each attribute pair and a pair of brackets indicate the beginning and ending of each definition record. Displayed below are some examples of input records for the dictionary:

```
[(anyone,anyone#1,pronoun#0)|gender,dual|count,single|countablity,single|pronoun
type,assertive]
[(anything,anything#1,pronoun#0)|gender,neutral|count,single|countablity,single|p
ronoun type,assertive]
[(aquarium,aquarium#0,container#0)|definition,no]
[(aquarium,aquarium#1,aquarium#0)|count,single]
[(aquariums,aquariums#1,aquarium#0)|count,plural]
[(are,are#1,be#0)|tense,present|person,second|count,single]
```

```
[(atoll,atoll#0,land mass#1)|definition,no]
[(atoll,atoll#1,atoll#0)|count,single]
[(atolls,atolls#1,atoll#0)|count,plural]
[(aunt,aunt#0,human#0)|definition,no]
[(aunt,aunt#1,aunt#0)|count,single|gender,female]
[(aunts,aunts#1,aunt#0)|count,plural|gender,female]
[(bachelor,bachelor#0,human#0)|definition,no]
[(bachelor,bachelor#1,bachelor#0)|count,single|gender,male]
[(bachelors,bachelors#1,bachelor#0)|count,plural|gender,male]
```

Example 3.1 - Input Records to the Dictionary

To create a dictionary entry the following steps are executed:

- First, a dictionary record is read in.
- Second, the record is parsed into a structure containing a natural language (NL) word, definition reference word, parent's definition reference word, and attribute pairs.
- Next, the first two characters of the NL word are used to retrieve the index into the first level of the lexicon.
- The lexicon is then searched using the NL word and the index from the previous step.
- If the lexicon entry is found, a definition structure node is created. The node is then positioned within the definition structure using the parent's definition reference word, to locate the node's position within the definition structure. The node is then connected to the lexicon by adding another definition pointer to the lexicon entry and having it point to the newly created node's location.
- If the corresponding first level lexicon entry is NULL then a lexicon entry node is created using the NL word. A definition structure is also created and connected to the appropriate lexicon entry.
- If the corresponding first level lexicon entry is not equal to NULL and there is one or more second level lexicon entries associated with the first level entry, then the new definition is first checked to see if it belongs in the first level of the lexicon. If it goes into the first level then the old first level entry is inserted at the front of the second level entry array. Otherwise, it is put at the end of the second level lexicon entry array and a sort is performed upon the lexicon entry array. After the new lexicon entry has been placed into the correct position in the lexicon, it's definition structure node is created and connected to the lexicon.
- If the corresponding first level lexicon entry is not equal to NULL and there is no second level lexicon entries associated with the first level entry, a new lexicon entry is created using the current input record. The new lexicon entry is compared to it's corresponding first level entry, the one that is alphabetically first, goes into the first level of the lexicon while the other entry goes into second level.
- After each definition node is loaded up into the definition structure, its attributes and attribute values are converted into attribute table indexes and are stored within the definition's attribute list. This saves space and attribute table access time.

3.4. Access

3.4.1. Lexicon

To retrieve a lexicon entry from the dictionary the following steps are performed.

- If the word used for the lookup is a definition word, it's "#" and identifying integer are removed to leave the NL word.
- Next, the first two characters of the NL word are used retrieve the appropriate index into the first level of the lexicon.
- The index from previous step is used to retrieve the first level lexicon entry.
- The NL word is compared to the first level lexicon entry word just retrieved. If a match, then the first level lexicon entry is returned.
- If the NL word didn't match the first level lexicon entry, a search is then performed using the lexicon entry's second level lexicon array. If a match, then the appropriate lexicon entry is returned otherwise a null value is returned.

3.4.2. Definition Structure

To retrieve a definition structure node, the following steps are performed.

- Perform the previous five steps from the preceding paragraph.
- If the returned lexicon entry is not equal to null value, a search is then performed upon the lexicon entry's definition pointers. This process uses the definition word as the key to retrieve a pointer to the appropriate definition structure node.

4. Attribute Table

4.1. Scope

- The attribute table is used to store attributes and their corresponding values so they can be used to define and differentiate word meanings.
- The attribute table is used in the following processes:
  - *Dictionary formation* - When definitions are loaded into the definition structure they reference the attribute table to retrieve indexes for attribute names and attribute values. This is to ensure quick referencing during the conceptualization process.
  - *Word sense parsing* - In the word sense parsing process the attribute table indicates if a definition node is a base word or a definition.
  - *Phrase parsing* - In the phrase parsing process the attribute table denotes different verb actions and count types.
  - *Pronoun resolution* - In the pronoun resolution process the attribute table indicates different noun genders and count types.
  - *Semantic net formation* - During the formation of the semantic net the attribute table indicates if a word is an "instance" and differentiates between pronouns and conjunction types.
  - *Normalization* - The attribute table in the normalization process indicates if adjectives are positive or negative.
- Sometimes attribute values represent other words stored in the definition structure for further referencing, negation, and classification.
- The attribute table, used in this experiment, is listed in appendix A.

4.2. Structure

- The attribute table consists of two data structures, the *attribute table* and the *attribute entry*.
- The *attribute table* contains an array of *attribute entry* pointers and an *attribute entry* counter. The *attribute table* can have up to 32,767 *attribute entries.*
- The *attribute entry* contains: a pointer to name, an attribute index , an array of character string pointers, and an *attribute entry* counter. There can be up to 32,767 attribute values for each *attribute entry*.
  - *Name pointer* - Stores the name of the attribute.
  - *Attribute index* - Holds the attribute's position in the attribute table.
  - *Array of character string pointers* - Stores the corresponding attribute values for each attribute.
  - *Attribute entry counter* - Stores the number of attribute values per attribute.
- An example of an attribute table is illustrated below.

Figure 4.1 - Example of the Attribute Table in Memory

4.3. Construction

The attribute table is the first structure to get loaded into memory in the conceptualization machine. Attribute input records have the following format:

`[attribute name, attribute value]`

- *Attribute name* - character string representing the name of the attribute.
- *Attribute value* - character string representing a value associated with an attribute.

Left and right brackets ("[ ]") are used to separate each attribute input record. Displayed below are some attribute table input records

```
[person,first]
[person,second]
[person,third]
[gender,male]
[gender,female]
[gender,dual]
[gender,neutral]
[gender,all]
```

Example 4.1 - Attribute table input records

To create an attribute table entry the following steps are executed:

- An input record is read from the attribute table file.
- The record is then parsed into an attribute name and an attribute value.
- Next, the *attribute table* is searched using the attribute name.
- If the attribute name is not found, a new *attribute entry* is created and is positioned at the end of the *attribute table*.
- The attribute value is then added to the *attribute entry's* list of values.
- After all the attribute table records have been read and stored in the *attribute table*, a sort performed upon the *attribute table* using the attribute name as the key.

4.4. Access

To retrieve an attribute entry, given an attribute name, the following is performed:

- A binary search is performed upon the attribute table using the attribute name as the key.
- When the attribute name is found, a numerical index is returned from the table indicating the position of the attribute entry.

To retrieve an attribute value index using an attribute name and attribute value, the following is performed:

- The attribute index retrieved from the previous paragraph is used to find the correct *attribute entry*. A linear search is then performed upon the attribute entry's array of attribute values to find the appropriate attribute value index.

4.5. Attribute Descriptions

Below is a list of attributes and their corresponding values defined in the conceptualization machine.

- *Action* - Indicates if a verb is a physical or momentary action, i.e. jump, run, etc. It has a yes or no value.
- *Caste* - Indicates if dictionary entry is alphabetic, alphanumeric, real number, or integer.
- *Collection* - Indicates the count ability of a noun to determine if it takes a plural or singular count. It can have the following values: mass, single, both, collective or part.
- *Count* - Indicates if a noun or pronoun is in singular, plural or both singular or plural form.
- *Definition* - Indicates if a definition node is used as a base word or as a definition. It can have the value of yes or no.
- *Description* - Indicates if a noun is either abstract or concrete; animate, inanimate, or location; edible; or negative.
- *Gender* - Specifies whether a noun or pronoun is a female, male, neutral, both female and male, or all possible combinations of the aforementioned.
- *Instance* - Indicates if a noun or a verb is a specific instance of a particular noun or verb. It can have the value of yes or no.
- *Name* - Indicates if a word is a proper noun. It can have the value of yes or no.
- *Negative Adjective* - Specifies an adjective with a negative meaning.
- *Negative* - Indicates if a word is negative. It can have the value of yes or no. If negative is not specified, the default value is positive.
- *Number type* - Indicates if a number is used as position, number, fraction, multiplier or a quantity.
- *Person* - Indicates whether a pronoun or verb is either in first, second or third person form.
- *Pronoun type* - Specifies if a pronoun is personal, reflexive, possessive, relative, interrogative, demonstrative, universal, assertive, or negative.
- *Specific* - This indicates if a noun can be used as a pronoun at times. The variable "specific" can have the values of yes or no. The value of "no" indicates that the noun can be used as a pronoun at times. The default value when "specific" is not specified is "yes".
- *Tense* - Specifies if a verb is in the past, present, or future tense.

5. Program Input

5.1. Description

The input processed by the program and used for testing college students was third-grade level science books about whales & dolphins, bears and spiders. These books were tested because of their:

- Lack of metaphors and other forms of literation, "Children's speech has an extreme literalness and absence of metaphor."[Chukovsky, 1963, p. 271]
- Simple sentence structure
- Small vocabulary
- Cognitively simple semantics
- Context is known
- Ease of use in identifying program errors
- Ease to measure program and student output
- Manageable quantity of words (between 400 and 900 words)

Another reason for using third-grade level texts is that they are the texts that children start to read and learn from. Hence, if a computer is to conceptualize natural language text and is starting with an empty knowledge base, it should also start with low level texts. Some excerpts of text used for this experiment is displayed below.

Whales and dolphins have fins and live in the sea, but they are not fish. They are mammals. They must rise to the surface of the water to breathe air. Whales, dolphins, and porpoises are members of the same animal family. Some of the family have teeth. Others do not. They hunt their food underwater. Blue whales can stay underwater for more than half an hour. They may be as heavy as twenty elephants. Blue whales can swim almost twenty miles per hour.

Example 5.1 - Text Excerpt Used in Machine Conceptualization

-

5.2. Assumptions

Some assumptions about the natural language text used in this experiment.

- The input text will be grammatically and syntactically correct.
- The input text will be absent of metaphors and other forms of literation. This will be explained in section 5.3.
- All words and character strings will be pre-defined in the program dictionary. If a word in the natural language input is not defined in the dictionary, it will be replaced with the word "text#0", a generic term. Generic terms are ignored in all the parsing processes and are not put into the semantic net.
- The vocabulary corresponds to the proper context of the text. This enables the program to use a simple *word sense* parser.
- The input text will contain no quotations, since double quotes are ignored by the parser.

- A blank line separates each paragraph.
- Compound words and word phrases that are longer than three words in length, will not be identified as such.

5.3. Natural Language Obstacles

There are many forms of literation that can cause tremendous difficulties in natural language processing. These obstacles are listed below with comments. They are also the reasons why third-grade level science books were chosen for input for this project.

- *Ambiguity* - Sentences usually have more than one meaning that humans can decipher immediately with ease. Machines on the other hand, experience difficulties in comprehending the meaning and sometimes fail altogether in deciphering the meaning of a word. [Touretzky, 1986, p. 212; Clark, Clark, 1977, p.80]
- *Analogies* - Interpreting analogies like: foot of a mountain; foot of list; mapping or projecting the human body onto to other things. [Lackoff, 1987, pp. 19-20]
- *Atypically* - Recognizing the relevance between properties or entities. [Touretzky, 1986, p. 27]
- *Changing definitions and standards* - Dealing with the problem of ever changing definitions and standards, such as stock market terms. [Sowa, 1984, p. 296]
- *Comparative Terms* - Interpreting terms such as good, poor, big, and small. Answering the questions: "How big is the sun?" or "How big is that spider?". [Sowa, 1984, p. 295]
- *Compounds* - "Many examples, i.e. topless bar, electrical engineer, ect., are what linguists call *compounds*. It is often the case that the meanings of compounds are not compositional; that is the meaning of the whole cannot be predicted from the meanings of the parts and the way they are put together." [Lackoff, 1987, p. 147]
- *Context* - The current situation, place, and time when something is said, have great amount of importance in deciding and understanding what was said. [Odell, 1981; Schank, 1975, pp. 16-17]
- *Emphasis* - The emphasis, de-emphasis and extra attention that humans place on particular words can change the meanings of words. [Odell, 1981]
- *Hedges* - The premodifiers put in front of words can modify and completely change the meaning of words greatly. Some examples of hedges are *strictly* speaking, *loosely* speaking, and *technically* speaking, where strictly, loosely, and technically are hedges. [Lackoff, 1987, p. 138]
- *Homonym* - A word with more than one meaning. [Clark, Clark, 1977, p. 444]
- *Idioms* - "Idioms are phrases with special meanings. Note that 'kick the dog' gets its sense from kick ('strike with ones foot') and dog ('canine animal'), for it means 'strike the canine animal with one's foot.' Not so with the idiom 'kick the bucket in the sense of 'die'. It does not mean 'strike the pail with ones foot.' to be represented, the phrase 'kick the bucket' must have a lexical entry all its own, one unrelated to those for kick and bucket. It is also true of other idioms, such as 'hit the sack', 'put one's foot in one's mouth', or be in hot water." [Clark, Clark, 1977, p. 446]
- *Interpretation* - "The totality of meaning is never fully rendered: there is an immense amount of implications, even in the most explicit of languages; or rather nothing is ever completely expressed, nothing exempts the subject who is listening from taking the initiative in giving an interpretation." [Merleau-Ponty, 1973, p. 29]

- *Lexical Creativity* - People keep on changing the language, making up new words or changing the meanings of old words when they can not find one for a particular situation. [Clark, Clark, 1977, pp. 446-447]
- *Metaphor* - A figure of speech that is applied to something to which it is not literally applicable in order to suggest a resemblance between the two, i.e. "Esther Williams is a regular fish." [Lackoff, 1987, p. 138]
- *Metonymy* - "Is the process of referring to something by the name of something else associated with it." [Sowa, 1991, p.10] For example: The White house lashed out at congress.
- *Nonfunctionality* - "What a given string of words means is not a function of the formal characteristics those strings possess. 'Why not?' can be used to make a request, even though its form is that of a question." [Odell, 1981]
- *Sincerity* - When we say something that is governed by the feelings we were expressing at the time. [Odell, 1981] For example: When a father tells his daughter's boyfriend "You touch her, you die", does not mean he is really going to kill his daughter's boyfriend.

These literation difficulties mentioned above are explained in more detail in appendix M.

6. Text Parsing

6.1. Input

The input to the *text parsing* step can either be a book stored in a file or sentences typed in by an user. An example of some input text is displayed below.

Whales and dolphins have fins and live in the sea, but they are not fish. They are mammals. They must rise to the surface of the water to breathe air.

Example 6.1 - Unparsed Paragraph

6.2. Process

The first thing to be done in conceptualizing natural language (NL) text is to read in the input text. Once read in, the text is stored in memory in a manipulative structure (*words'* data structure) for easy processing. This is done by reading the text in one character at a time and identifying it as either an alphanumeric character, space, period, comma, question mark, end of line, or a double quote. The different character types trigger a variety of program actions. Listed below are the actions they trigger.

- *Alphanumeric character* - The character is added to a temporary character string and the alphanumeric character counter is incremented by one.
- *Space* - If the alphanumeric character counter is greater than zero, the temporary character string is added to the *words* structure and the alphanumeric character counter is set to zero.
- *Period* - The temporary character string is added to the *words* structure, the word "period" is also added to the *words* structure, the alphanumeric character counter is set to zero, and a new sentence is started in the *words* data structure dependent upon the next character read in.
- *Comma* -
  - If the current character is numeric then another character is read in. If the new character is also numeric, the previous character is added to the temporary character string and the alphanumeric character counter is incremented by one. Otherwise, the temporary character string is added to the *words* structure, the word "comma" is also added to the *words* structure, and the alphanumeric character counter is set to zero.
  - If the current character is non-numeric, then the temporary character string is added to the *words* data structure, the word "comma" is also added to the *words* structure, and the alphanumeric character counter is set to zero.
- *Question mark* - The temporary character string is added to the *words* data structure, the word "question" is also added to the *words* structure, the alphanumeric character counter is set to zero, and a new sentence is started in the *words* structure dependent upon the next character read in.
- *End of line character* - A new paragraph is added to the *words* structure dependent upon the next character read in.
- *Double quote* - No action is taken.

After all the characters have been read in, the *words* structure is checked for compound words and word phrases consisting up to three words. If any are found using the dictionary, they are then

condensed into a one word element. When compound nouns are encountered and the last word is a noun, then a sentence is added to the end of the current paragraph in the form of "*compound noun is a noun*". This is done to enhance the categorization process.

6.3. Output

The output from the *text parsing* process is a *words* data structure that contains an interconnected lists of words, sentences, and paragraphs. An example of the *words* structure is displayed below.

Figure 6.1 - Sample Words Data Structure

7. Word Sense Parsing

7.1. Input

The input to the *word sense* parsing process is the *words* data structure, describe in the previous chapter. A sample of the *words* data structure is displayed in figure 6.1.

7.2. Process

The *word sense* parser uses only grammar and syntax to determine which meaning of a word is to be assigned to the natural language word read in. It is assumed that the context of the input text is known and that there are no metaphors or other forms of literation. Metaphors and other forms of literation can to complicate the parsing process. The *word sense* parsing algorithm listed below determines a natural language word's sense in linear time.

```
1)  A word is read in.
2)  The lexical entry of the word is found in the dictionary.
3)  If there is only one definition for the lexical entry, that sense of the
    word is then chosen. Return to step (1).
4)  If there are two definitions and if one of them is not to be used as a
    definition, then the other definition is chosen as the word sense to be
    used and the program returns to step (1). Otherwise the program goes to
    step (5).
5)  Until the first correct sense of the word is found, check each definition
    in the lexical entry. This step is expanded in the paragraph below.
6)  Return to step (1).
```

Algorithm 7.1 - Word Sense Parser

To find the correct sense of the word, the following tests are performed in the order displayed. Each definition contained within the lexical entry found in step 2 is tried until the proper definition is found. The following logic is executed.

If the grammar of the current definition is a:

- *Preposition* -
  - If the grammar of the previous word was a noun or pronoun and the grammar of the following word could possibly be a noun, pronoun, adjective, determiner, or a premodifier then return the current definition.
  - If the grammar of the previous word was a comma and the grammar of the following word could possibly be an adjective, determiner, or an adverb then return the current definition.
  - If the grammar of the previous word was an adjective and the grammar of the following word could possibly be an adjective, determiner, or a premodifier then return the current definition.
  - If the grammar of the previous word was a determiner and the grammar of the following word could possibly be a premodifier or a noun then return the current definition.
  - If the grammar of the previous word was a conjunction and the grammar of the following word could possibly be a determiner then return the current definition.

- If the grammar of the previous word was a verb and the grammar of the following word could possibly be an adjective or a determiner then return the current definition.
- If the grammar of the previous word was an adverb and the grammar of the following word could possibly be an adjective, determiner, premodifier or a noun then return the current definition.
- If its the first word in the sentence and the grammar of the following word could possibly be an adjective, noun or a premodifier then return the current definition.

- *Adjective* -
  - If the grammar of the following word could possibly be a noun or an adjective then return the current definition.
  - If the grammar of the following word could possibly be a pronoun and the grammar of the previous word was a determiner then retrieve the next definition. If there isn't another definition then return the current definition.
  - If the grammar following the current word could possibly be some sort of syntax and the grammar of the previous word was a noun or an adverb then return the current definition.
  - If the grammar of the following word could possibly be a preposition and the grammar of the previous word was an adjective or a verb then return the current definition.
  - If the grammar of the following word could possibly be a premodifier and the grammar of the previous word was a conjunction then return the current definition.
- *Verb* -
  - If the grammar of the following word could possibly be an auxiliary and if the grammar of any of the words following the auxiliary could possibly be a verb then retrieve the next definition.
  - If the grammar of the following word could possibly be a verb and the grammar of the current word could possibly be an auxiliary then retrieve the next definition.
  - If the grammar of the following word could possibly be a verb and the grammar of the previous word could possibly be a noun or pronoun then return the next definition.
  - If the grammar of the following word could possibly be a negative and the grammar of the word following the negative could possibly be a verb then retrieve the next definition else return the current definition.
  - If the grammar of the following word could possibly be a noun, adjective or a determiner then return the current definition.
  - If the grammar of the following word could possibly some kind of syntax and the grammar of the previous word was a noun or pronoun then return the current definition.
  - If the grammar of the following word could possibly an adverb and grammar of the word following the adverb could possibly be a verb then retrieve the next definition.
  - If the grammar of the following word is a premodifier then return the current definition.
  - If the grammar of the previous word was an auxiliary and the grammar of the following word is not a verb then return the current definition.
  - If the grammar of the previous word was a pronoun and the grammar of the following word could possibly be a preposition then return the current definition.

- If the grammar of the current word could possibly be an auxiliary and the grammar of any of the words following it could not possibly be a verb then return the current definition.

- *Noun* -
  - If the grammar of the previous word was an auxiliary and the grammar of the current word could be a verb then retrieve the next definition.
  - If the grammar of the previous word was a negative, the grammar of the word before that was an auxiliary and the grammar of the current word could possibly be a verb then retrieve the next definition.
  - If the grammar of the following word could possibly be a pronoun and the grammar of the word following that one could possibly be a verb or an auxiliary then return the current definition. Otherwise retrieve the next definition.
  - If the grammar of the following word could possibly be a noun, verb, auxiliary, conjunction, preposition, or some kind of syntax then return the current definition.
- *Determiner* -
  - If the grammar of the following word could possibly be a noun, adjective or pronoun then return the current definition.
  - If the grammar of the following word could possibly be a preposition and the grammar of the previous word was a preposition then return the current definition.
- *Premodifier* -
  - If the grammar of the following word could possibly be a noun, adjective, determiner or preposition then return the current definition.
- *Adverb* -
  - If the grammar of the following word could possibly be a verb then return the current definition.
  - If the grammar of the following word could possibly be a pronoun and the current word is the first word in the sentence then return the current definition.
  - If the grammar of the following word could possibly be a determiner and the grammar of the previous word a noun then return the current definition.
- *Pronoun* -
  - If the grammar of the following word could possibly be a noun then retrieve the next definition.
  - If the grammar of the following word could possibly be a verb, adverb, auxiliary, or some kind of syntax then return the current definition.
  - If the grammar of the following word could possibly be a preposition, the grammar of the previous word was a determiner and the grammar of the word before the previous word was a preposition then retrieve the next definition, otherwise return the current definition.
  - If the grammar of the following word could possibly be a determiner and the grammar of the previous word was a verb then return the current definition.
  - If the grammar of the following word could possibly be some kind of syntax and the grammar of the previous word was a preposition then return the current definition.
  - If the grammar of the following word could possibly be a conjunction and the grammar of the previous word was a preposition then return the current definition.

- *Conjunction* -
  - Return the current definition.
- *Interjection* -
  - Return the current definition.
- *Negative* -
  - Return the current definition.
- *Auxiliary* -
  - Retrieve the next definition if the grammar of the following word could be a verb; the grammar of any words after that could possibly be a pronoun, syntax, preposition, conjunction, or noun; and a verb has not been encountered. Return the current definition if a verb has been encountered.
  - If the grammar of the following word could possibly be a verb and the grammar of the previous word was a pronoun then return the current definition.
  - If the grammar of the following word could possibly be a noun, adjective, verb, or auxiliary then return the current definition.
  - If the grammar of the following word could possibly be a negative or an adverb and the grammar of the word after that is a verb then return the current definition.
  - If the grammar of the following word could possibly be a pronoun or a preposition and the grammar of the word after that is a verb then return the current definition.
- *Syntax* -
  - Return the current definition.

7.3. Output

The output of the *word sense parsing* process is the same as the *words* data structure used as input, except that the NL words have been replaced with dictionary reference words. Displayed below is an example of the *word sense parsing* process output.

Figure 7.1 - Example of Word Sense Parsing Output

8. Book Structure Construction

8.1. Purpose

The book structure construction step transforms the *words* structure into an indexed *book structure*. This is done so that all the paragraphs, sentences, and words can be referenced with both ease and speed, since they will be stored in indexed arrays.

8.2. Input

Input to the *book structure construction* process is the *words* data structure created in the *word sense parsing* routine explained in the previous chapter.

8.3. Structure

The *book structure* consists of three interconnected objects, the *book*, *paragraph*, and *sentence* structures. Each of the structures reference the dictionary. The *book* data structure is a dynamic array of *paragraph* pointers that can reference up to 32,767 paragraphs. The *paragraph* data structure is a dynamic array of *sentence* pointers that can reference up to 32,767 sentences. The *sentence* data structure is a dynamic array of *dictionary entry* pointers that can reference up to 32,767 *dictionary entries*. The *book structure* can reference up to 35 trillion dictionary entries or about a billion sentences. *Book*, *paragraph*, and *sentence* structures are displayed in detail in appendix H.

8.4. Process

The process of transforming the *words* data structure into the *book structure* is straight forward. The transformation algorithm is displayed below. The *words* data structure is referenced in the below algorithm as just ***words***.

```
    Set the paragraph, sentence, and word pointers equal to the head of the    words data
structure.
    While the words paragraph pointer is not equal to NULL do
        Add a new paragraph to the book structure.
        While the words sentence pointer is not equal to NULL do
            Add a new sentence to the book structure.
            While the words word pointer is not equal to NULL do
                Retrieve the address to the appropriate dictionary entry using
    the word pointer's character string.
                Insert the dictionary entry address into the book structure.
                Set the words word pointer equal to the word pointer's next
                word.
            End While
            Set the words word pointer equal to the sentence pointer's next
    sentence.
            Set the words sentence pointer equal to the sentence pointer's
            next sentence.
        End While
        Set the words word pointer equal to the paragraph pointer's next
        paragraph.
```

```
        Set the words sentence pointer equal to the paragraph pointer's next
    paragraph.
        Set the words paragraph pointer equal to the paragraph pointer's next
    paragraph.
    End While
```

Algorithm 8.1 - Book Structure Construction

The *Book Structure Construction* algorithm transforms the *words* data structure into the *book* data structure. This is done quite easily, since the *words* data structure contains paragraph pointers, sentence pointers, and word pointers that correspond to the *book* structure's paragraph, sentence and dictionary entry pointers. The algorithm contains the following logic. Whenever the algorithm encounters a null *words* sentence pointer it creates a new paragraph in the *book* data structure. Whenever the algorithm encounters a null *words* word pointer it creates a new sentence in the *book* data structure. This continues until all *words* paragraph, sentence, and word pointers are equal to null.

8.5. Output

An example of how the *book* data structure is stored in the computer's memory is displayed below.

Figure 8.1 - Example Book Structure

9. Phrase Parsing

9.1. Purpose

The purpose of the *phrase parsing* process is to break up sentences into simple verb phrases. This is done to:

- Ease the transformation of natural language text into a semantic net.
- Simplify pronoun resolution.
- Identify subjects (actors), actions, objects of the action, subject modifiers, action modifiers, object modifiers, phrases that modify subjects, and phrases that modify objects within each sentence.

9.2. Input

Input to the *phrase parsing* process is the *book structure* detailed in the previous chapter.

9.3. Structure

9.3.1. Phrase Node

The *phrase structure* to be built in the *phrase parsing* process consists of five objects. Each phrase node consists of a(n):

- *Subject Pointers* - A dynamic array of node pointers that references subject(s) of the verb phrase.
- *Action Pointers* - A dynamic array of node pointers that references action(s) of the verb phrase.
- *Subject Counter* - An integer that accumulates the number of subjects in a verb phrase.
- *Action Counter* - An integer that accumulates the number of actions in a verb phrase.
- *Originating Sentence Number* - An integer that stores the number of the sentence, that the verb phrase originated from. This number is referenced in the construction of complex concepts.

The above format allows for one or more actors to perform one or more actions upon zero to many objects. It also allows the identification of complex situations that consist of many actors and actions. A representation of the phrase node in memory, with its corresponding subject and action nodes, is displayed below.

Figure 9.1 - Phrase Node

9.3.2. Subject Node

The subject node has several pointers that reference information about the subject of the verb phrase. They are the following:

- *Subject* - A definition pointer that references the subject of the phrase in the *definition structure*. This pointer labels the subject.

- *Premodifier* - A dynamic array of definition pointers that identify the subject's premodifiers.
- *Post-Modifier* - A dynamic array of definition pointers that identify the subject's post-modifiers.
- *Phrase Structure Pointer* - A pointer to another *phrase structure* that modifies the subject.
- *Premodifier Quantity* - An integer that stores the number of the subject's premodifiers.
- *Post-Modifier Quantity* - An integer that stores the number of the subject's post-modifiers.

The subject node can store almost any kind of subject configuration that can be found in natural language. This is possible because the subject has pointers that reference pre and post modifiers and other modifying phrases.

9.3.3. Action Node

The action node also has several pointers that reference information about the action of the verb phrase. They are the following:

- *Action* - A definition pointer that references the action of the phrase in the *definition structure*. This pointer labels the action of the phrase.
- *Premodifier* - A dynamic array of definition pointers that identify the action's premodifiers.
- *Post-Modifier* - A dynamic array of definition pointers that identify the action's post-modifiers.
- *Object Pointers* - A dynamic array of subject node pointers that references the object(s) of the action.
- *Premodifier Quantity* - An integer that stores the number of the action's premodifiers.
- *Post-Modifier Quantity* - An integer that stores the number of the action's post-modifiers.
- *Object Quantity* - An integer that stores the number of objects of the action.

The action node can store almost any kind of action and object configuration that can be found in natural language. This is possible because the action node has pointers that can reference many modifiers. In addition, the action node has an array of pointers that can reference any number of action objects.

The *phrase node*, *phrase section*, and *phrase root* are interconnected objects that make up the *phrase structure*. The *phrase root* is made up of a dynamic array of *phrase section* pointers and *phrase section* counter. The *phrase section* consists of a dynamic array of *phrase node* pointers and a *phrase node* counter. The phrase section is used to represent the paragraph structure of the original NL text. A representation of phrase structure is displayed below.

Figure 9.2 - Phrase Structure

The c++ objects that store and reference information about the subject of the phrase, action of the phrase, phrase node, phrase section, and phrase root are listed in appendix H.

9.4. Process

To create the phrase structure, a fifty state automaton is used to transform the book structure into verb phrases. The automaton's transition from state to state is determined by the current word's grammar and the order that the grammar is in. Other actions that are executed in the automaton are determined by

the word's count, the word's gender and vocabulary. The algorithm used for phrase parsing is elaborated below.

```
    For each paragraph in the book structure do
        Add a new phrase section to the phrase root.
        For each sentence in the current paragraph do
            Input the sentence into the first state(Q0) of the automaton.
        End For
    End For
```

Algorithm 9.1 - Phrase Parsing

A small example is given here, to explain how the phrase parse automaton works. The sentence that will be parsed into a phrase is "*whales are mammals*". The current word is initialized to the first word of the sentence, that is "*whales*" for this example. The sentence first enters the automaton starting at state "Q0". The state "Q0" queries the grammar of the current word in the sentence. The grammar of the word "*whales*" is a noun, so the automaton enters into its subject state "Q4". State "Q4" puts "*whales*" into the subject slot of the phrase node and sets the current word to the next word in the sentence that is "*are*". State "Q4" queries the grammar of the current word in the sentence. The grammar of the word "*are*" is a verb, so the automaton enters into the action state "Q22". State "Q22" puts "*are*" into the action slot of the phrase node and sets the current word to the next word in the sentence that is "*mammals*". State "Q22" queries the grammar of the current word in the sentence. The grammar of the word "*mammals*" is a noun, so the automaton enters into the object state "Q25". State "Q25" puts "*mammals*" into the object slot of the phrase node and sets the current word to the next word in the sentence. The next word is set equal to null. When the null pointer is reached the phrase parse automaton is exited.

A brief description of what each state does and the flow of the automaton and can be found in appendix C.

9.5. Output

A textual representation of the phrase parsing process is displayed below. Shown first is the input sentence followed by the corresponding *phrase structure*.

```
*** Section 1 ***

whales#0 and#0 dolphins#0 have#7 fins#0 and#0 live#1 in#0 the#0 sea#1 comma#0 but#0 they#1 are#4
not#0 fish#1 period#0

Phrase
    Subject : whales#0
    Subject : dolphins#0
    Action  : have#7
    Object  : fins#0
    Action  : live#1
    Object  : sea#1
               Pre-Modifiers
              in#0
                  the#0

whales#0 and#0 dolphins#0 have#7 fins#0 and#0 live#1 in#0 the#0 sea#1 comma#0 but#0 they#1 are#4
not#0 fish#1 period#0

Phrase
    Subject : they#1
```

```
    Action  : are#4
               Post-Modifiers
                  not#0
    Object  : fish#1

they#1 are#4 mammals#0 period#0

Phrase
    Subject : they#1
    Action  : are#4
    Object  : mammals#0
```

Example 9.1 - Text Representation of a Phrase Section

10. Pronoun Resolution

10.1. Purpose

The purpose of the pronoun resolution process is to eliminate ambiguity. The computer tries to resolve all the non-specific nouns, pronoun references within the *phrase structure*, and clarify all "do" actions. When the *phrase structure* is transformed and inserted into the *semantic net*, the sentence and paragraph structure of the original input text will be lost. This will cause non-specific nouns, pronouns and "do" verbs to become too ambiguous for processing, since their previous references will not be accessible as they were in their natural language format. To elaborate further, non-specific nouns and pronouns that stand alone in the semantic net have no reference to what they were. The same thing can be said about the verb "do". It implies an action is being executed, but that action is unknown, if it is standing alone. In summary, the pronouns and "do" verbs should be replaced with their referencing noun and verb counterpart before the *phrase structure* is processed into the *semantic net.*

10.2. Input

Input to the *pronoun resolution* process is the *phrase structure* described in the previous chapter.

10.3. Process

The *pronoun resolution* process is executed at the phrase section level. It is done at this level, since pronouns are dependent upon which paragraph and sentence they are located in. Pronoun resolution resolves phrase parts in the following order: non-specific noun subjects and non-specific noun objects, pronoun subjects, pronoun objects, modifying concept pronoun subjects, null objects, and "do" actions.

10.3.1. Non-Specific Noun Subject & Object Resolution

Non-specific subject nouns are resolved by the following logic:

```
If the first post-modifier of the subject is equal to "of" then
    If the following post-modifier of the subject is also a noun then
        If the counts of the non-specific noun and the post-modifier noun
are compatible then:
            Replace the non-specific subject noun with the post-modifier
noun.
        End If
    End If
End If
```

Non-specific object nouns are resolved by the following logic:

```
If the first post-modifier of the object is equal to "of" then
    If the following post-modifier of the object is also a noun then
        If the counts of the non-specific noun and the post-modifier noun
are compatible then:
            Replace the non-specific object noun with the post-modifier
noun.
```

```
            End If
        End If
    End If
```

10.3.2. Subject Resolution

Subject pronoun resolution is done by comparing the previous phrase's subject gender and count against the current pronoun. If they are equivalent the pronoun subject is replaced with the previous phrase's subject. If the gender and count are not equivalent in the previous sentence and "this" is a pre-modifier of the pronoun, then the pronoun is replaced with the subject of the previous phrase. If the pronoun is still unresolved, and the previous phrase's subject and object are both pronouns and if the object has a modifying phrase that has a subject that has a noun, then the current subject's pronoun is replaced with the noun.

10.3.3. Object Resolution

Object pronoun resolution is more complex than subject resolution. If the object's pronoun is reflexive then the current phrase's subject replaces the pronoun object. Nothing is done if the object pronoun is assertive. If no resolution has occurred yet, then the machine compares the previous phrase's object gender and count against the pronoun's gender and count. If they are equivalent, the machine replaces the pronoun with the previous object. If the pronoun is still not resolved and the previous phrase's subject gender and count are compatible with the pronoun's gender and count, then the object pronoun is substituted with the previous subject.

10.3.4. Subject of the Modifying Phrase Resolution

The modifying phrase subject's pronoun is resolved in one of the following ways: If the pronoun is demonstrative, relative, or is equal to "their" and it's gender and count matches that of the main phrase's object then the pronoun is substituted by the main phrase's object. If the pronoun is still unresolved and the main phrase's subject gender and count are compatible with the pronoun's gender and count, then the pronoun is replaced with the main phrase's subject. Otherwise the previous phrase's subject gender and count are compared with the pronoun's gender and count. If they are equivalent then the pronoun is replaced by the previous phrase's subject.

10.3.5. Null Object and "do" Actions Resolution

Null objects and "do" actions are run through the resolution process to be resolved. If the null object has an adjective for a premodifier then the adjective is put into the missing object's slot. Otherwise if there is more than one object and the previous object's pre-modifier is an adjective then the previous object's pre-modifier is used for the object. If the object is still unresolved and there is more than one action, then the previous action's object is used to fill in the null object. If all else fails and the action has an auxiliary premodifier then the object slot is filled by the action and the action is replaced with the auxiliary premodifier. If the action is equal to "do" then the action is replaced with the previous phrase's action.

10.4. Output

Output of the *pronoun resolution* process is just a modification of the incoming *phrase structure*. An example of some pronoun resolved phrases are displayed below with their corresponding input sentences.

```
*** Section 1 ***

whales#0 and#0 dolphins#0 have#5 fins#0 and#0 live#1 in#0 the#0 sea#1 comma#0 but#0 they#1 are#4
not#0 fish#1 period#0

Concept
   Subject : whales#0
   Subject : dolphins#0
   Action  : have#5
   Object  : fins#0
   Action  : live#1
      Post-Modifiers
         in#0
   Object : sea#1
      Pre-Modifiers
         the#0

Concept
   Subject : whales#0
      Pre-Modifiers
         they#1
   Subject : dolphins#0
   Action  : are#4
      Post-Modifiers
         not#0
   Object  : fish#1


they#1 are#4 mammals#0 period#0

Concept
   Subject : whales#0
      Pre-Modifiers
         they#1
   Subject : dolphins#0
   Action  : are#4
   Object  : mammals#0
```

Example 10.1 - Textual Representation of Sample Phrase Nodes (after pronoun resolution)

11. Semantic Net Formation

11.1. Purpose

The following reasons are why a *semantic net* was used to store information gathered from natural language text:

- It is an established memory configuration that has been used for many years.
- Up to 32,767 relations can created per entity in the semantic net used in this experiment.
- The semantic net can grow to an infinite size, limited only by system memory.
- A hierarchical structure can be built within the net to promote the categorization of entities.
- Relations and entities can be moved up, down, and across the net as the need arises.
- Any entity can have a relation with any other entity within the net.
- Many forms of logic can be applied and executed upon the semantic net.
- The structure of the semantic net enables the identification of *basic-level* entities.
- With the net's entity and relation structure, concept importance and *best example* calculations can performed.
- Complex concepts can be constructed within the semantic net.
- Inheritance of relations between different levels of entities is built into the net.
- Concepts are better described and clarified when they can be related to one another as they are within a semantic net.

With the reasons listed above, the semantic net was the ideal structure to be used for the conceptualization of natural language text.

11.2. Input

The input to the *semantic net formation* process is the pronoun resolved *phrase structure* described in the previous chapter.

11.3. Structure

The structure of the semantic net is a hierarchical unary tree that uses interconnected *entity* and *relation* data structures. To simplify the description of the semantic net, a *relationship* will be defined as two entity nodes connected by one relation node or one entity node connected with one relation node.

11.3.1. Entity Node

The *entity* node is used to represent the subject or object of the *phrase structure*, minus their modifiers. The entity node consists of the following relevant parts with their corresponding descriptions:

- *Character String* - The string stores the entity label.
- *Entity Quantity* - An integer that stores the number of relations emanating from the entity.

- *Relation Pointers* - A dynamic array of pointers that references the entity's relations.
- *Entity Pointers* - A dynamic array of pointers that references entities that the entity has relations with.
- *Distance* - The number of levels from the semantic net's root node.
- *Instance Switch* - Indicates if the entity is an instance of another entity or not.

The cardinality between the phrase structure's subject and object nodes and the semantic net's entity nodes is many to one. This is due to storing information about many subject nodes and object nodes of the same kind into one entity node. Compressing the data into the semantic net enables more information and implications to be made about a particular entity. That is, if someone said "*those five boxes are red*" and "*the three boxes over there are green*" then it could be implied that boxes can be both red and green. The semantic net's ability to condense information into one entity allows for many implications. Format of the entity node can be found in appendix H.

11.3.2. Relation Node

The *relation* node represents the action data that was contained in the *phrase structure*. Listed below are the relevant parts of the relation node along with their corresponding descriptions:

- *Character String* - A character string that labels the relation.
- *Relation Pre-Modifiers* - A dynamic array of character strings that stores the action's premodifiers.
- *Relation Post-Modifiers* - A dynamic array of character strings that stores the action's post-modifiers.
- *Subject Pre-Modifiers* - A dynamic array of character strings that stores the corresponding subject's pre-modifiers.
- *Subject Post-Modifiers* - A dynamic array of character strings that stores the subject's post-modifiers.
- *Object of the Relation Pre-Modifiers* - A dynamic array of character strings that stores the object's pre-modifiers.
- *Object of the Relation Post-Modifiers* - A dynamic array of character strings that stores the object's post-modifiers.
- *Concept Number* - A number used to logically connect many entities and relations together, to represent a complex concept.
- *Negation Switch* - An integer that indicates that the relation has the opposite meaning of what is represented by the relation's name. For example, if the relation "is" had its negation switch turned on then the relation would mean "isn't".
- *Moved Integer* - Indicates how far a relation has been moved up or down within the semantic net from it's originating starting position.

The cardinality between the phrase structure's action nodes and the semantic net's relation nodes is one to one. The exception to this rule is identical relationships, then the cardinality can be many to one. The reason for a one to one cardinality and the storage of subject and object modifiers along with the relation node is that events usually describe actions that happen at specific points in time about entities in a particular state. Format of the relation node can be found in appendix H and the system defined relations used in this experiment can be found in appendix E.

11.3.3. Semantic Net

11.3.3.1. Formal Definition

A semantic net is a knowledge representational model that has the following characteristics:

- Nodes are connected by labeled directed arcs that represent binary relations.
- It is acyclic.
- It usually has a class hierarchy.
- Nodes inherit properties from their superordinate classes. Typically "ISA" relations are used to defined semantic net hierarchies.
- Nodes can represent physical objects, situations, events, actions, sets, attributes, etc..
- Nodes also can represent classes of objects, objects, or instances of an object.
- Every concept is represented by a unique node or a set of nodes.
- Its purpose is to represent concepts expressed in natural language sentences and phrases.
- There are no unconnected nodes.
- A node can have 0 to n arcs emanating from it.

[Biebow, Chaty, 1993; Brachman, 1979; Harris, 1985; Hendrix,1978; Hendrix, 1979; Rumelhart, 1972; Shapiro, 1991; Simmons, 1973; Sowa, 1984; Winston, 1992]

11.3.3.2. Semantic Net Project Definition

The semantic net used in this project has the characteristics listed above plus the following features:

- Entities (nodes) can have relations with other entities not only directly below them, but with entities many levels above them, below them, and across from them.
- Relations within the net are directional.
- Each entity has only one primary parent, but can have any number of secondary parents. Primary parent relations are labeled with "ISA", while secondary parent relations are labeled with "isa".

11.4. Process

Many routines and algorithms are involved in encapsulating the *phrase structure* data into the semantic net. The following tasks and algorithms are performed and executed to accomplish this transformation process.

- First the semantic net must be initialized with the entity root node, labeled "entity".
- Second, the *AddPhraseToNet* algorithm is executed for every combination of subject, action and object within every verb phrase to insert the phrases into the semantic net. This algorithm triggers the following two algorithms to perform the following tasks:
    - *GetEntityFromNet Algorithm* - The algorithm tries to locate the given subject identifier with the semantic net. If the identifier is absent from the net, the algorithm will then create an entity labeled with the given identifier and insert it into the appropriate position in the net.
    - *GetEntityFromNet Algorithm* - The object identifier must be located within the semantic net. If it is absent from the net, then an object entity must be created and positioned in the correct position within the net.

- After the subject and object have been inserted into the semantic net the *AddRelation* algorithm is executed to perform the following tasks:
  - The action of the phrase is translated and transformed into the appropriate relation node.
  - The subject entity is connected to the object entity using the relation node that was created in the previous step.

The aforementioned steps are discussed in greater detail in the following sections.

11.4.1. AddPhraseToNet Algorithm

The following algorithm separates each verb phrase from the *phrase structure* to be processed by the *AddPhraseToNet* algorithm.

```
    Initialize sentence counter to 0.
    For each phrase section in the phrase structure do
        For each phrase node in the current phrase section do
            If current phrase is from the same sentence as the last phrase,                then
decrement the sentence counter by 1.
            Execute AddPhraseToNet(sentence counter, current phrase node).
            Increment sentence counter by 1.
        End For
    End For
```

Algorithm 11.1 - Verb Phrase Separation

The *AddPhraseToNet* algorithm initiates the construction of a relationship for every combination of subject, action and object from the verb phrase. This needs to be done, since there can be more than one subject or object per a phrase and an entity node can only store one subject or object. In addition to that, for every action a relation must be created within the semantic net. Displayed below is the *AddPhraseToNet* algorithm.

```
    For each subject in the verb phrase do
        Get grammar of the current subject.
        If the grammar of the subject is a noun then
            subject entity = GetEntityFromNet(subject, subject pre-modifiers)
            For each action of the phrase do
                For each object of the action do
                    AddRelationship(subject entity, phrase, action number,
                    object number)
                End For
            End For
        End If
    End For
```

Algorithm 11.2 - AddPhraseToNet

11.4.2. GetEntityFromNet Algorithm

The *GetEntityFromNet* algorithm can be summed up with the following logic. If the subject or object can not be found in the semantic net, then create its entity node in the proper position within the net. To find a subject within the semantic net, the subject is compared with every entity node that can be reached by traversing the "Isa", "Has Subset", and "Instance" relations starting from the net's root. The first match between the subject identifier and entity node identifier is found then the entity node is returned. Otherwise the missing entity node is created and added to the semantic net. The following steps are executed to create an entity node within the semantic net.

- An entity node is created using the subject or object identifier.

- An "isa" relation is created to connect the new entity to the root of the semantic net, unless the subject is an instance of a particular object then an "instance of" relation is created instead.
- An "includes" relation is created to connect the root of the semantic net to the new entity, unless the subject is an instance of a particular object then an "instance" relation is created instead.
- If the verb phrase's subject or object has any noun or adjective pre-modifiers, then the "has subset" and "subset of" relations must be created to reference the subject or object. The subset relations are then used to connect the semantic net root to the newly created entity node and visa versa. For every noun and adjective modifier the subject entity has, it must be further defined down in the semantic net. One level is added for each one of the subject's noun and adjective modifiers, in descending order, to the newly created entity. For example, If there was a "*large gray elephant*" subject then the following relationships would be needed (relations are highlighted in bold, modifiers are underlined and each indentation represents another level lower within the semantic net):

  entity ***includes*** *elephant*
  elephant ***isa*** *entity*
  elephant ***has subset*** *gray elephant*
  gray elephant ***subset of*** *elephant*
  gray elephant ***has subset*** *large gray elephant*
  large gray elephant ***subset of*** *gray elephant*

11.4.3. AddRelation Algorithm

The *AddRelation* algorithm executes many routines to add relationships into the semantic net.

- It first uses a routine to translate the action identifier into a *relation name*. The *relation name* could be a system defined *relation name* like "isa", "isnota", "includes", "is", "have", "havenot", and "isnot" or a dictionary entry identifier.
- If the grammar of the object is a noun then:
  - Its corresponding entity node is retrieved from the semantic net.
  - Otherwise, if the *relation name* is equal to "is", the grammar of the object is an adjective, the adjective ends in "est", and its post-modifier is a noun, then the *relation name* is changed to "isa", the adjective is made into pre-modifier of the object, and the post-modifying noun is made into the object.
- If the *relation name* is equal to "isa" then the subject and object are connected with "isa" and "includes" relations, following the rule that the subject entity may only have one primary parent.
- If the *relation name* is equal to "includes", then the subject and object roles are reversed and are connected together with "isa" and "includes" relations while adhering to the rule that the previous object may have only one primary parent.
- If the *relation name* is equal to "instance of", then the subject and the object are connected together with "instance of" and "instance" relations.
- Add the appropriate pre and post modifiers to the newly created *relation node*.

11.5. Output

Textual and graphical examples of a semantic net are displayed below. "A" is an abbreviation for *actor*, "R" is an abbreviation for *relation* and "O" is an abbreviation for *object* of the relation.

```
Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=fin#0
       0.0  A=entity  R=INCLUDES  O=sea#0
       0.0  A=entity  R=INCLUDES  O=fish#0
       1.0  A=entity  R=INCLUDES  O=mammal#0
       2.0  A=entity  R=INCLUDES  O=surface#0
       2.0  A=entity  R=INCLUDES  O=water#0
       2.1  A=entity  R=INCLUDES  O=air#0

Entity = fin#0
   Relations :
       0.0  A=fin#0  R=ISA  O=entity

Entity = sea#0
   Relations :
       0.0  A=sea#0  R=ISA  O=entity

Entity = fish#0
   Relations :
       0.0  A=fish#0  R=ISA  O=entity

Entity = mammal#0
   Relations :
       1.0  A=mammal#0  R=ISA  O=entity
       1.0  A=mammal#0  R=INCLUDES  O=whale#0
       1.0  A=mammal#0  R=INCLUDES  O=dolphin#0

Entity = whale#0
   Relations :
       0.0  A=whale#0  R=HAVE  O=fin#0
       0.0  A=whale#0  R=live#1 in#0  O=sea#0
       0.0  A=whale#0  R=ISNOTA  O=fish#0
       1.0  A=whale#0  R=ISA  O=mammal#0
       2.0  A=whale#0  R=rise#1 to#0  O=surface#0 of#0, water#0, to#0,
       2.1  A=whale#0  R=breathe#1  O=air#0

Entity = dolphin#0
   Relations :
       0.0  A=dolphin#0  R=HAVE  O=fin#0
       0.0  A=dolphin#0  R=live#1 in#0  O=sea#0
       0.0  A=dolphin#0  R=ISNOTA  O=fish#0
       1.0  A=dolphin#0  R=ISA  O=mammal#0
       2.1  A=dolphin#0  R=breathe#1  O=air#0
       2.0  A=dolphin#0  R=rise#1 to#0  O=surface#0 of#0, water#0, to#0,

Entity = surface#0
   Relations :
       2.0  A=surface#0  R=ISA  O=entity

Entity = water#0
   Relations :
       2.0  A=water#0  R=ISA  O=entity

Entity = air#0
   Relations :
       2.1  A=air#0  R=ISA  O=entity
```

Example 11.1 - Textual Semantic Net

Figure 11.1 - Graphical Semantic Net

12. Semantic Net Modification

There are four different types of semantic net manipulations that make the net more manageable and less complex. They are called:

- *Entity Generalization*
- *Multiple Inheritance Resolution*
- *Entity Specification*
- *Relationship Reduction*

The above modifying processes will determine an entity's primary parent, generalize attributes up the net, makes sub-category names more descriptive, and combine duplicate categories into one. Each of these processes is describe in more detail below.

12.1. Entity Generalization

12.1.1. Purpose

Entity generalization promotes repetitive attributes within a category up the semantic net. This process will compress the semantic net that will result in a saving of system memory. In addition, the identification and differentiation between categories will be faster and easier.

12.1.2. Process

The entity generalization process starts at the bottom of the semantic net and works its way up to the root of the net using recursion. The process searches each node in the net for *generic attributes*. A *generic attribute* is an attribute for which all the children of a parent entity have the identical relationship (attribute). For example, in figure 11.1, the relationship "HAVE Fins" is a *generic attribute*, since both the whale and the dolphin have that relationship. When a *generic attribute* is found, it is added to the parent entity and removed from all the parent's children. Looking at figure 11.1, this would mean remove the "HAVE Fins" relationship from both the whale and dolphin entities and attaching it to the mammal entity. This generalization process was done to the scenario pictured in figure 11.1 and the end result of the process is pictured in figure 12.1, displayed below. Looking at the two figures,11.1 and 12.1, several of the relationships that both the whale and dolphin entities had were promoted up to the mammal entity.

Figure 12.1 - Semantic Net After Relation Inference

12.2. Multiple Inheritance Resolution

12.2.1. Purpose

During the process of transforming and inserting *phrase structures* into the semantic net, entities within the net might have develop multiple inheritance. This is fine, unless an entity's primary parent is not the most descriptive parent that an entity it could have. For example, if an entity had two parents and the secondary parent is at a lower level than the primary parent then the secondary parent is assumed to be more descriptive than the primary parent. If the previous example is true, then the *multiple*

inheritance resolution process would rectify it by making the most descriptive parent, the parent at the lower level, the primary parent. This process is referred to as *Primacy Validation & Rectification.*

Another situation that arises during the formation of the semantic net is that an entity can foster several parents within the same path in the net. To conserve memory the net uses, the duplicate relations to the related parents are deleted. This will be referred to as *Inherent Multiplicity Elimination*. Displayed below is part of a semantic net that has both invalid primacy and duplicate inheritance. An example of invalid primacy for the "*blue whale*" are the relationships "*mammal INCLUDES blue whale*" and "*whale includes blue whale*" relations pictured in figure 12.2 below. An example of duplicate inheritance for the "*blue whale*" are the two paths "*blue whale isa whale*" along with "*whale ISA mammal*" and "*blue whale ISA mammal*".

Figure 12.2 - Semantic Net with Invalid Primacy and Duplicate Inheritance

The Primacy Validation & Rectification and Inherent Multiplicity Elimination algorithms are discussed in detail below.

12.2.2. Primacy Validation & Rectification

Invalid primacy is identified when an entity has both "isa" and "ISA" relations and the "isa" relation points to an entity that is at lower level than the entity that the "ISA" relation points to. If the previous situation is identified then it can be rectified by switching the offending "isa" relation with the entity's "ISA" relation and switching the corresponding offending parent's "includes" relation with the other parent's "INCLUDES" relation. The result of this process applied to the semantic net represented in figure 12.2, is displayed below in figure 12.3. It can be observed that the "*blue whale isa whale*" relationship was changed to "*blue whale ISA whale*" and that the "*blue whale ISA mammal*" relationship was changed to "*blue whale isa mammal*".

Figure 12.3 - Semantic Net with Valid Primacy

12.2.3. Inherent Multiplicity Elimination

Multiple inheritance is detected when an entity has two or more "isa" paths that eventually lead to the same ancestor entity whose level is greater than zero. Example of this condition is displayed in figure 12.3. When this situation occurs, then the "isa" relation connected to the highest level entity is eliminated along with its corresponding "includes" relation. Displayed below is the semantic net represented in figure 12.3 after it has gone through *Inherent Multiplicity Elimination*. It can be observed in figure 12.4, that the "*blue whale isa mammal*" relationship was removed from the net along with its corresponding includes relation "*mammal includes blue whale*".

Figure 12.4 - Semantic Net after the Inherent Multiplicity Elimination

12.3. Entity Specification

12.3.1. Purpose

The *entity specification* algorithm is used modify to sub-category names to be more descriptive.

12.3.2. Process

The *entity specification* algorithm is a two stage process. It must first identify sub-categories that can be described with better modifiers. Second it then modifies the pre-modifiers with a suitable descriptor. A modifiable sub-category can be identified by three different characteristics. First, it is a subset of another category. Second, its modifiers don't contain an adjective. Third, it contains a relationship where the relation is labeled with "*HAVE*" or "*HAVE NOT*" and the object of the relationship is an actual physical part of the subject entity. For example, the relationship "*whale HAVENOT teeth*" displayed in figure 12.5.

After the aforementioned sub-category has been identified, several steps are taken to make it more descriptive. First, the appropriate positive or negative adjective (dependent upon if the relation was equal to *"HAVE" or "HAVENOT"*) is retrieved from the attribute table. The key to the attribute table is the object body part, in this instance "*TOOTH*" with a positive or negative indicator. The value retrieved from the table is then added to the subject's list of premodifiers, i.e., "*toothless*". Figure 12.5 displays the semantic net before the entity specification process and figure 12.6 shows net after the process.

Figure 12.5 - Semantic Net Before the Entity Specification

Figure 12.6 - Semantic Net After the Entity Specification

12.4. Relationship Reduction

12.4.1. Purpose

Relationship reduction is a two stage process that is executed to eliminate duplicate sub-categories. Eliminating duplicate sub-categories the process shrinks the semantic net.

12.4.2. Routine 1

The relationship reduction process consists of two routines. Each routine is called recursively starting its modifying action from the bottom of the semantic net. *Routine 1* looks for an entity with a pair of duplicate relationships where the relation is equal to "*HAS SUBSET*" and the objects of both relations are the same (i.e., WHALE has two relationships of "*HAS SUBSET toothless WHALE*" displayed above in figure 12.6). If this is the case, then all the relationships of the second subset are added to the first subset and the second subset entity is removed from the semantic net. Displayed below in figure 12.7, is the result of the *Routine 1* applied to the semantic net shown in figure 12.6. Observe in figure 12.7 that there is only one "toothless whale" entity left.

Figure 12.7 - A Semantic Net After Relationship Reduction (Routine 1)

12.4.3. Routine 2

The second routine, *Routine 2*, looks for two types of relationship sets with the following criteria:

- Criterion one has the following characteristics that can be found below in figure 12.8.
    - A subject entity that has a relation equal to "HAS SUBSET".

- The object entity excluding its premodifiers, of the "HAS SUBSET" relation is equal to the subject part of the relationship.
- The premodifier of the object is either a positive or negative adjective representing a physical part of the object.
- The object entity has a relation that is equal to "HAVE" or "HAVENOT" and the object of that relationship is equal to the corresponding physical part of the pre-modifying adjective mentioned in the previous sentence.
- For example, the relationships "*WHALE HAS SUBSET toothless WHALE*" and "*toothless WHALE HAVE BALEEN*".

- Criterion two has the following characteristics that can be found below in figure 12.8.
  - The subject entity that has a relation that is equal to "INCLUDES".
  - The object of the "INCLUDES" relation mentioned in the previous sentence is equal to the subject.
  - The aforementioned object has a relationship where the relation is equal to "HAVE" or "HAVENOT" and the object of the relation is a physical part of the of the aforementioned object.
  - For example, the relationships "*WHALE INCLUDES WHALE*" and *"WHALE HAVENOT TOOTH*".

If the two previously stated criteria are true, then the relationships of the object entity of the second relationship are added to the object entity of the first relationship and the object of the second relationship along with all of its relationships are removed from the semantic net. Figure 12.8 shows a semantic net with the above criteria. Figure 12.9 is the result of *Routine 2* applied to the semantic net shown in figure 12.8.

Figure 12.8 - Semantic Net with Excess Relationships

Figure 12.9 - A Semantic Net After Relationship Reduction (Routine 2)

12.5. Testing The System's Conceptualization Logic Capabilities

Nineteen sets of sentences were processed by the machine conceptualization process to test the machine's logic capabilities. The results and input of these nineteen tests are listed in appendix F. The sets of sentences ranged from easy to complex logic situations.

13. Entity Categorization

13.1. Introduction

The categorization is a major tool involved in thinking. It embodies the perception of body movement, learning, and memory. It enables the description of groups of objects in the environment. Categorization is a part of conceptualization recognized since the time of Aristotle. It renders hierarchies of rank in biology, institutions, government, science and business. Categorization also helps people to predict underlying similarities and differences among objects. Categorization plays a large role in the human cognitive process. If a machine has the ability to categorize then a good measure of a machine's conceptualizing facilities can be determined by how well it can form a taxonomy.

Categorization is a very important facet of the conceptualization process for the following reasons:

- Conceptual systems are organized in terms of categories. Most, if not all of our thought involves those categories. [Lackoff, 1987]
- Rosch and others have observed that categories in general have *best examples* (*called prototypes*) and that all of the specifically human capacities like body movement, perception, mental image formation, learning, memory, and other body functions play a role in categorization. [Lackoff, 1987, p. 7]
- Our perception of the world is divided into categories and is illustrated in our languages. Searle [1978] the philosopher embellishes upon this by saying:

  > "I am saying that what counts as reality - what counts as a glass of water or a book or a table, what counts as the same glass or a different book or two tables - is a matter of the categories that we impose on the world; and those categories are for the most part linguistic. And furthermore; when we experience the world we experience it through linguistic categories that help to shape the experiences themselves. the world doesn't come to us already sliced up into objects and experiences: what counts as an object is already a function of our system of representation, an how we perceive the world in our experiences is influenced by that system of representation. .... Our concept of reality is a matter of our linguistic categories." [p. 184]

- Smith and Medin [1981] elaborate that people classify concepts into categories and draw relations between and within those categories:

  > "Most would agree that people use concepts both to provide taxonomy of things in the world and to express relations between classes in that taxonomy. The taxonomic function can itself be split into the following two generally agreed upon component functions: 1. Categorization - This function involves determining that a specific instance is a member of a concept (i.e. this particular creature is a guppy) or that one particular concept is a subset of another (i.e. guppies are fish)." [p. 7]

- Gelman [1988] claims that children use categorization as a inferencing tool:

  > "Young children use category membership to predict underlying similarities among objects - even when perceptual similarity would lead to a different prediction." [p. 66]

- Back to the time of Aristotle, categorization has been recognized as a conceptualization tool.

  > "Since the time of Aristotle, philosophers have argued that our conceptual system is articulated by a core of ontologically basic categories, such as physical object and event. Within the category of physical objects, living thing is such a category, as are animal and plant within the category of living things." [Carey, 1985, p. 162]

- Berlin, Breedlove, and Raven [1974] state our existence is dependent upon categorization:

  > "Man is, by nature, a classifying animal. His continued existence depends, in fact, on his ability to recognize similarities and differences among objects and event in his physical universe and to mark these similarities and differences linguistically. " [p. 25]

- Bruner, Goodnow, and Austin [1956] explain that we would be overcome by the complexity of the world around us if it wasn't for our ability to categorize:

  > "But were we to utilize fully our capacity for registering the differences in things and to respond to each event encountered as unique, we would soon be overwhelmed by the complexity of our environment. Consider only the linguistic task of acquiring a vocabulary fully adequate to cope with the world of color differences! The resolution of this seeming paradox -- the existence of discrimination capacities which, if fully used, would make us slaves to the particular -- is achieved by man's capacity to categorize. To categorize is to render discriminatory different things equivalent, to group objects and events and people around us into classes, and to respond to them in terms of their class membership rather than their uniqueness." [p. 1]

- Simmons and Correira [1979] summarize that we transform literature into a hierarchical conceptual structure after reading it:

  > "A series of psychological studies ([Kintsch 1974] [van Dijk 1975] [Meyer 1975] [Thorndike 1977] and [Crothers 1972]) offers strong evidence for the hypothesis that a person, after reading a text, has developed a macro structure that organizes the propositions of the text in a hierarchical form. The propositions highest in the hierarchy are more general than those that occur at the lower levels." [p. 363]

- Pirsig [1974] embellishes on the idea that our society is based upon concept hierarchies:

  > "This structure of concepts is formally called a hierarchy and since ancient times has been a basic structure for all Western knowledge. Kingdoms, empires, churches, armies have all been structured into hierarchies. Modern business are so structured. Tables of contents of reference material are so structured, mechanical assemblies, computer software, all scientific and technical knowledge is so structured - so much so that in some fields such as biology, the hierarchy of phylum-order-class-genus-species is almost an icon." [p.93]

- Mazlack in his paper exploring the necessity of concept formation, relates that categorization is a part of conceptualization:

  > "The process of conceptualization is that of progressive category formation."

Since categorization is important and essential to conceptualization, as iterated above, it must be shown to what degree a machine method can categorize a body of natural language text.

13.2. Purpose

The goal of this chapter is to determine to what extent a machine method can categorize a set of entities. This is done by demonstrating the degree of similarity between a machine method and a human, in classifying a set of entities. The demonstration includes:

- Illustrating and reviewing the process involved in classification
- Explicating the machine method's categorization output
- Discussing the student survey categorization questions
- Explaining the measures used to evaluate the machine method's performance
- Validating the machine method's output using the student survey results

- Elaborating upon the student survey result inconsistencies

With the previous steps executed, the quality and human similarity of the machine categorization method can be determined to some degree.

13.3. Process

The process of constructing and updating a taxonomy within the semantic net happens during the transformation and merging of the *phrase structure* into the net. It happens at this stage because the semantic net is built around a hierarchy and the input text determines how entities are to be classified within that hierarchy. Happening at this stage, the categorization process requires a complex algorithm due to the unwieldy nature of natural language input text. Being unwieldy means the text does not have a fixed structure, contains repetitive attributes for each entity, possess numerical data for the most part, or have an identifiable attribute set. Another factor that adds to the complexity of the algorithm is the infinite number of word combinations it will encounter during the categorization process. Displayed below is the skeletal view of the categorization algorithm from the *Semantic Net Formation* section.

```
For each phrase do
     For each subject in the phrase do
          For each action of the phrase do
               For each object of the action do
                    AddRelationship(subject, action, object, phrase)
               End For
          End For
     End For
End For
```

The *AddRelationship* procedure, displayed above, uses the selected phrase's subject(s), action(s), and object(s) combinations to build relationships. It then tries to update the taxonomy, within the semantic net, with the newly created relationships using the following logic:

- Whenever any physical or abstract entity is encountered and can not be found in the semantic net, it becomes a semantic net root child. This is done by connecting the child node (*dog*) to the root node with an "ISA" arc and the root node to child (*dog*) node with an "INCLUDES" arc.

Figure 13.1 - Before Dog Has Been Inserted into the Net

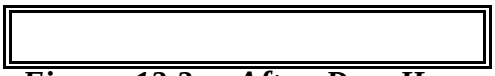

Figure 13.2 - After Dog Has Been Inserted into the Net

- If the relation of the current entity (*entity 1*) indicates that *entity 1* is a member of another entity (*entity 2*) and *entity 2* is located at a lower level than the current entity's parent (*entity 3*), displayed below in figure 13.3, then the following is done:

Figure 13.3 - Sample Semantic Net before Manipulation

- The "ISA" and "INCLUDES" arcs between *entity 1* and *entity 3* are changed to "isa" and "includes" arcs respectively.

Figure 13.4 - Primary Relations Are Changed to Secondary Relations

- An "ISA" relation is added from entity 1 to entity 2 and an "INCLUDES" relation is added from entity 2 to entity 1.

Figure 13.5 - Entity 2 Becomes Entity 1's Primary Parent

- If the relation of the current entity (entity 1) indicates that entity 1 is a member of another entity (entity 2) and entity 2 is located at a higher level than the current entity's parent (entity 3), displayed in figure 13.6, then the following is done:

Figure 13.6 - Sample Semantic Net Before Manipulation

- An "isa" relation is added from entity 1 to entity 2 and an "includes" relation is added from entity 2 to entity 1.

Figure 13.7 - Entity 2 Becomes a Secondary Parent of Entity 1

- A subset relation between two entities is determined by the pre-modifiers of the entities and the relation between them. For example, the sentences "*Some elephants are circus elephants*" and "*Elephants are big gray things*" indicate "circus elephants" is a subset of "elephants", "gray things" is subset of "things", and "big gray things" is a subset of "gray things". The following actions are executed with the sentence, "*some elephants are white*", since it contains subset indicators.

Figure 13.8 - Semantic Net Before Manipulation

- The entity node "some elephant" is created.
- A "SUBSET OF" arc is added from "*some elephant*" to "*elephant*" and a "HAS SUBSET" arc is added from "*elephant*" to "*some elephant*".

Figure 13.9 - Semantic Net After Manipulation

- When an entity node is labeled with a proper name and the relation is some form of "be", it indicates that particular entity is an instance of some other entity. If the current entity (Joe) is a proper name and it is a member of another entity (human), i.e., "*Joe is a human*", then the following is executed:

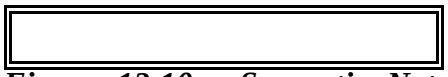

Figure 13.10 - Semantic Net Before Manipulation

- An "INSTANCE OF" arc is added from "Joe" to "human" and a "INSTANCE" arc is added from entity "human" to "Joe". The results of these actions are illustrated in the figure below.

Figure 13.11 - The Entity Joe Becomes a Primary Instance of Human

- If a primary instance relation has already been established between "Joe" and "student", then a "instance of" arc is added from "Joe" to "human" and an "instance" arc is added from "human" to "Joe". This is to indicate a secondary instance relationship.

Figure 13.12 - Semantic Net Before Manipulation

Figure 13.13 - Semantic Net After Manipulation

The above methods, only illustrate the "main" logic of the categorization process since there are many ways an entity can be classified dependent upon the content of the input text. Located in appendix F are the results of 19 tests that were used to evaluate the integrity of the above categorization methods. The results indicated that the machine categorization method could interpret logically difficult, conflicting, and diverse paragraphs of text.

13.3.1. Other Categorization Efforts

AI categorization efforts listed below, are not applicable to the current project since these efforts need an identifiable attribute set, demand a strict input format, are not all incremental, require repetitive attributes (*i.e., "A dog has four legs and hair", "A cat has four legs and hair", and "A cow has four legs and hair"*), and require numerical data (*not true for all*). The classification method used in this project had to be sensitive to the content of the input text and be able to react accurately to it. Reacting accurately means to analyze not only the subjects and objects of a phrase, but also the relations, and pre and post modifiers contained within the phrase. The categorization method used in this experiment had to classify entities by literal content of the input and not by repetitive attributes.

- Clustering algorithm - *Agglomerative Hierarchical Clustering* [Everitt, 1993; Fisher, Pazzani, 1991]
- Clustering algorithm - *Centroid Clustering* [Everitt, 1993]
- Clustering algorithm - *Group Average Clustering* [Everitt, 1993]
- Clustering algorithm - *Nearest Neighbor Clustering* [Everitt, 1993]
- ER clustering algorithm - *Entity-Relationship Clustering* [Rauh, Stickel, 1992]
- Hierarchical classification - *Instance Partition Method* [Martin, Billman, 1991]
- Hierarchical classification - *Spreading Activation Method* [Smith, Medin, 1981]
- Incremental classification - *ID5R* [Utgoff, 1989; Ragavan, Rendell, Shaw, Tessmer, 1993]
- Non-Incremental classification - *ID3* [Quinlan, 1986; Thorton, 1992; Utgoff, 1989]
- Partitioning classification - *Bayesian Analysis* [Anderson, Matessa, 1991]
- Pattern composition classification - *ID4* [Utgoff, 1989]
- Semantic net clustering - *Conceptual Clustering* [Cheng, Fu, 1985]
- Unsupervised clustering - *COBWEB and OXBOW* [Iba, Gennari, 1991; Reich, Fenves, 1991]
- Unsupervised clustering - *CYRUS and UNIMEM* [Fisher, Pazzani, 1991]
- Unsupervised clustering - *EPAM* [Fisher, Pazzani, 1991]
- Version space learning algorithm - *Candidate Elimination* [Thorton, 1992]
- Version space learning algorithm - *Most General or Specific Subsumption* [Woods, 1991]

13.4. Output

The result of the machine categorization method is a structure representing a hierarchy of entities that includes category memberships, subsets, and instances. The hierarchy structure can be displayed in

two different ways: *indented text column hierarchy display* and a *semantic net blue print*. The two types of displays are illustrated and discussed in detail below.

13.4.1. Indented Text Column Hierarchy Display

The most generic hierarchy display produced by the machine method, is the *indented text column hierarchy display* that illustrates visually parent and child relationships without their corresponding relation labels. It has the following configuration:

- The hierarchical entities are listed in a vertical column.
- The top most entity sitting farthest to the left is the root of the hierarchy.
- An entity's parent is the first entity that is both above it and is one tab to the left of it when going from bottom to top.
- An entity's children are those entities displayed both below it (*before another entity at the same level is displayed*) and one tab to the right of it.
- An entity is a member of any category that is both above it and two or more tabs to the left of it.
- An entity is member of the first category that is both above it and is one tab to the left of it when going from bottom to top.

The *indented text column hierarchy display* does not indicate relation types, i.e., whether an entity is a member, subset, or instance of another entity. Shown in figure 13.1 is an *indented text column hierarchy display* generated by the machine method that was given the following natural language input.

Whales are mammals.
Some whales have teeth.
Dolphins are mammals.
Some dolphins do not have teeth.
Porpoises are mammals.
Porpoises have teeth.

```
root entity
   mammal#0
      whale#0
         toothed#0 whale#0
      dolphin#0
         toothless#0 dolphin#0
      porpoise#0
   tooth#0
```

Example 13.1 - Indented Text Column Hierarchy Display

Referring to figure 13.1, the following information can be gleaned:

- The "mammal#0" and "tooth#0" entities are members and children of the root entity.
- The "whale#0", "dolphin#0", and "porpoise#0" entities are members and children of the "mammal#0" entity. This can also be rephrased as to say that whale, dolphin, and porpoise entities belong to the mammal category.
- The "toothed#0 whale#0" entity is a member and child of the "whale#0" entity. This can also be rephrased to say that the "toothed#0 whale#0" entity belongs to the "whale#0" category.
- The "toothless#0 dolphin#0" is a member and child of the "dolphin#0" entity. This can also be rephrased to say that the "toothless#0 dolphin#0" entity belongs to the "dolphin#0" category.

- It also can be said that "whale#0", "toothed#0 whale#0", "dolphin#0", "toothless#0 dolphin#0", and "porpoise#0" entities are all members of the "mammal#0" category either directly or indirectly.

The *indented text column hierarchy display* is used to illustrate categorization results in a generic visual way to easily identify categories and sub-categories within the semantic net. It will also be used to evaluate the accuracy of the machine classifications.

13.4.2. Semantic Net Blue Print

The *semantic net blue print* displays all the entities within the net along with their relationships in a top down left to right order. "A" indicates actor, "R" indicates relation, and "O" indicates object of the relation. Entities that belong within the taxonomy are those that originate from the "includes", "INCLUDES", "instance", "INSTANCE" and "HAS SUBSET" relations. Displayed below is the *semantic net blue print* created from the natural language text displayed in the previous section. The mammal's parent is the entity indicated by an "ISA" relation. Its children are the whale, dolphin, and porpoise entities, that are indicated with the "INCLUDES" relation.

```
Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=mammal#0
       1.0  A=entity  R=INCLUDES  O=tooth#0

Entity = mammal#0
   Relations :
       0.0  A=mammal#0  R=ISA  O=entity
       0.0  A=mammal#0  R=INCLUDES  O=whale#0
       2.0  A=mammal#0  R=INCLUDES  O=dolphin#0
       4.0  A=mammal#0  R=INCLUDES  O=porpoise#0

Entity = whale#0
   Relations :
       0.0  A=whale#0  R=ISA  O=mammal#0
       1.0  A=whale#0  R=HAS SUBSET toothed#0  O=whale#0

Entity = toothed#0 whale#0
   Relations :
       1.0 toothed#0  A=whale#0  R=SUBSET OF  O=whale#0
       1.0 toothed#0  A=whale#0  R=HAVE  O=tooth#0

Entity = dolphin#0
   Relations :
       2.0  A=dolphin#0  R=ISA  O=mammal#0
       3.0  A=dolphin#0  R=HAS SUBSET toothless#0  O=dolphin#0

Entity = toothless#0 dolphin#0
   Relations :
       3.0 toothless#0  A=dolphin#0  R=SUBSET OF  O=dolphin#0
       3.0 toothless#0  A=dolphin#0  R=HAVENOT  O=tooth#0

Entity = porpoise#0
   Relations :
       4.0  A=porpoise#0  R=ISA  O=mammal#0
       5.0  A=porpoise#0  R=HAVE  O=tooth#0

Entity = tooth#0
   Relations :
       1.0  A=tooth#0  R=ISA  O=entity
```

Example 13.3 - Semantic Net Blue Print

13.4.3. Bear Article Machine Categorization Indented Hierarchy

The structure displayed below, is part of the category hierarchy built by the machine's categorization method after it had been given an article about bears to process.

```
entity
   bear#0
      brown bear#0
         biggest#0 brown bear#0
            kodiak bear#0
         fiercest#0 brown bear#0
            grizzly bear#0
      blue bear#0
      black bear#0
         moon bear#0
      mother#1 bear#0
      spectacled bear#0
      sloth bear#0
         mother#1 sloth bear#0
      smallest#0 bear#0
         sun bear#0
      polar bear#0
```

Figure 13.14 - Machine Categorization Output (Bear Article)

The above machine categorization output indicates the following. All the words listed below "entity" and are indented once to the right of it are considered members of "entity", such as "bear#0". The bears listed underneath "bear#0" and indented once to the right of it are members of the "bear#0" category, such as "brown bear#0", "blue bear#0", "black bear#0", etc.. This categorization methodology is also true for the rest of the entities pictured above.

13.4.4. Whale Article Machine Categorization Indented Hierarchy

The structure displayed below, is part of the category hierarchy built by the machine's categorization method after it had been given an article about whales and dolphins to process.

```
entity
   mammal#0
      whale#0
         toothed#0 whale#0
            largest#0 toothed#0 whale#0
               sperm whale#0
            orca whale#0
               killer whale#0
         toothless#0 whale#0
         blue whale#0
         humpback whale#0
         sperm whale#0
         small#0 whale#0
      dolphin#0
         toothed#0 dolphin#0
         toothless#0 dolphin#0
   water#0
   air#0
   animal#0
```

```
   porpoise#0
      toothed#0 porpoise#0
      toothless#0 porpoise#0
```

Figure 13.15 - Machine Categorization Output (Whale Article)

The above machine categorization output indicates the following. Whales and dolphins are members of the mammal category. Toothed whales, toothless whales, blue whales, etc., are members of the whale category. Largest toothed whale and orca whale are members of the toothed whale category.

13.4.5. Spider Article Machine Categorization Indented Hierarchy

The structure displayed below, is part of the category hierarchy built by the machine's categorization method after it had been given an article about spiders to process.

```
entity
   arachnid#0
      spider#0
         house spider#0
         garden spider#0
         hammock spider#0
         male#0 spider#0
         malmignette#0
            female#1 malmignette#0
         black widow#0
            American#0 black widow#0
         mother#1 spider#0
         wolf spider#0
         trapdoor spider#0
            funnel-web#0
         water spider#0
         jumping spider#0
            zebra spider#0
               little#0 zebra spider#0
         crab spider#0
         bird-eating spider#0
   insect#0
   creature#0
```

Figure 13.16 - Machine Categorization Output (Spider Article)

The above machine categorization output indicates the following. The spider is a member of the arachnid category. The house spider, garden spider, hammock spider, etc., are members of the spider category. Arachnid, insect, and creature are entities.

13.5. Student Categorization Survey

The medium used to validate the machine categorization method was a survey given to a class of college students. In the survey, five categorization questions were asked of the students after they had read an article about entities that they were to classify. Each question contained a selected entity followed by five categories that the entity could possibly be a member of. If the student believed that the selected entity belonged to a category they were to circle that category, otherwise they would not. The format of the categorization questions is displayed in the example below.

*Use the knowledge you learned from reading the previous article to answer the following questions. If you are **not** sure of the category, **don't** circle it.*

X) Circle the letter of the following category(s) that include entity 1.
- *a. category 1*
- *b. category 2*
- *c. category 3*
- *d. category 4*
- *e. category 5*

Example 13.3 - Categorization Question Format

After the student survey was handed in, the categorization questions were consolidated into numerical data so they could be used to validate the machine categorization results. The example question above was converted into numerical data by transforming it into a series of five integers with the values of either 1 or 0. A value of 1 was given if the student circled the category or a value of 0 if they did not. For example, using the above sample question, if the student circled categories 1 and 3 then the transformed result would be the numerical series (1, 0, 1, 0, 0). The numerical conversion was done for all five survey questions that resulted in array of twenty-five integers, with the values of 1 or 0. This was done for each participating student. The results were then compared to the machine's categorization results. The student and machine categorization results are located in *appendix I*.

13.6. Validation of Machine Output

In the absence of a standardized testing instrument, measuring expert performance is difficult. Classically, human experts measure whether some new person is also an expert by how consistent the new person is with the recognized experts. A new person can be thought of as an expert if his performance is consistent with expert performance to the degree that the experts' performances are consistent with each other.

The objective of the following test is to measure to what extent a machine method can categorize objects. One way of determining this measure is by comparing the machine's *dual-group-consistency score* to the *dual-group-consistency score* of a class of students. The comparison is used to determine if the computer's category selections were similar to a class of students' category selections. This was done by comparing the computer's category selections to each student in the class and comparing each student's selections with every other student's selections in the class.

A *dual-group-consistency score* was developed for the machine. This was formulated by first, comparing the machine category selections to each student's category selections in the class. This resulted in an array of similarity scores. The array dimension was the size of the class. The similarity scores were then aggregated into one *dual-group-consistency score* by averaging them together. The idea being to discover how consistent the machine's category selections were with the selections of the students considered as a group. The consistency score indicated how similar the machine was to the student aggregrate in categorizing objects. A high average indicates a high level similarity.

A *dual-group-consistency score* was developed for the students as a group. This was done by first, comparing each student's category selections with every other student's selections, excluding duplicate pairings. This resulted in array of similarity scores. The array had a dimension of n(n-1)/2, where n is the number of students in the class. The similarity scores for the students were then aggregated into one *dual-group-consistency score* by averaging them together. The idea being to discover how generally consistent the students were in their category selections amongst themselves.

The extent that a machine could categorize objects was measured by comparing the machine's *dual-group-consistency score* with the class's *dual-group-consistency score*. The scores were considered the same if they differed by no more than a tolerance value. In this experiment, the tolerance value was the variance of the class *dual-group-consistency score*.

13.6.1. Similarity Score

The similarity score between two sets of category selections was measured by calculating the Pearson (Phi) Correlation Coefficient of the two. The Pearson statistic was used for two reasons. First, it indicates if two variables are similar to one another. Second, it is the appropriate correlation coefficient to use when the two variables being compared are two valued.

The Pearson coefficient indicates similarity and dissimilarity between two variables with the following indicators. A Pearson coefficient with a value close to one indicates a high level of similarity between the two variables, a value near zero indicates little or no similarity between the two variables, and a value close to negative one indicates variables that go in opposite directions. The formula for the Pearson Correlation Coefficient (similarity score) is detailed below.

, where

n is the number of categories that can be selected and
x and y can be either machine/student selections or student/student selections.
[Mosteller, Rourke, 1973]

13.6.2. Machine Categorization Evaluation Sample

To show how the machine method and class *dual-group-consistency score*s were derived an example is calculated below:

- First, five students are given an article about bears to read.
- Second, each student answered the question below and four others like it (T*he other four questions asked the students to categorize a Kodiak Bear, Moon Bear, Grizzly Bear and Sloth Bear using the same categories listed below*).

4) Circle the letter of the following category(s) that include polar bears. *a. brown bears* *b. bears* *c. black bears* *d. flesh eaters* *e. biggest brown bears*

Example 13.4 - Categorization Question

- The results to the above question are displayed below:
  - One student circled brown bears.
  - Five students circled bears.
  - Zero students circled black bears.
  - Five students circled flesh eaters.
  - Zero students circled biggest brown bears.
- The machine method is then given the same article that the students had read and it produced the *indented text column hierarchy display*, shown below.

```
entity
   bear#0
      brown bear#0
         biggest#0 brown bear#0
            kodiak bear#0
         fiercest#0 brown bear#0
            grizzly bear#0
      blue bear#0
      black bear#0
         moon bear#0
      mother#1 bear#0
      spectacled bear#0
      sloth bear#0
         mother#1 sloth bear#0
      smallest#0 bear#0
         sun bear#0
      polar bear#0
   eater#0
      flesh#0 eater#0
         polar bear#0
```

Example 13.5 - Indented Text Column Hierarchy Display

- Using the machine's *indented text column hierarchy display*, it can be determined that the machine method classified the polar bear as a bear and flesh eater, but not as a brown bear, black bear or biggest brown bear.
- The machine method and student replies are then converted into numerical data that are displayed in the grid below. A "1" means the particular category was selected and a "0" means the particular category was not selected.

		Machine	Students				
Entity	**Category**	**m**	**s1**	**s2**	**s3**	**s4**	**s5**
Polar Bear	brown bear	0	0	1	0	0	0
	bear	1	1	1	1	1	1
	black bear	0	0	0	0	0	0
	flesh eater	1	1	1	1	1	1
	biggest brown bear	0	0	0	0	0	0
Kodiak Bear	brown bear	1	1	1	1	1	0
	bear	1	1	1	1	1	1
	black bear	0	0	0	0	0	0
	flesh eater	0	0	0	0	0	0
	biggest brown bear	1	1	1	1	1	1
Moon Bear	brown bear	0	0	1	0	0	0
	bear	1	1	1	1	1	1
	black bear	1	1	1	1	1	1
	flesh eater	0	1	1	1	1	0
	biggest brown bear	0	0	0	0	0	0

Grizzly Bear	brown bear	1	1	1	1	1	0
	bear	1	1	1	1	1	1
	black bear	0	0	0	0	0	0
	flesh eater	0	0	1	0	0	0
	biggest brown bear	0	0	0	0	0	0
Sloth Bear	brown bear	0	0	1	0	0	0
	bear	1	1	1	1	1	1
	black bear	0	0	0	0	0	0
	flesh eater	0	0	0	1	0	0
	biggest brown bear	0	0	0	0	0	0

Table 13.1 - Sample Categorization Numerical Replies

- The machine's responses are then correlated with the responses of all five students using the Pearson Correlation Coefficient, to compute the machine's similarity score. Each student's responses are then correlated with every other student, excluding any duplicate pairings, to compute the student's similarity scores. The result of these correlations is displayed in the table below:

	Machine	**Students**				
Students	**m**	**s1**	**s2**	**s3**	**s4**	**s5**
s1	0.92					
s2	0.67	0.72				
s3	0.85	0.92	0.62			
s4	0.92	1.00	0.72	0.92		
s5	0.84	0.77	0.56	0.71	0.77	

Table 13.2 - Categorization Similarity Scores Example

- The machine's *dual-group-consistency score* is computed by averaging together the correlation coefficients in column "m" (second column). The class's *dual-group-consistency score* is computed by averaging together the correlation coefficients in columns "s1" through "s5". The tolerance value for the class's *dual-group-consistency score* is calculated by taking the variance of columns "s1" through "s5". The result of these three operations is displayed in the table below.

	Machine	**Students**
Mean	0.84	0.77
Variance		0.02

Table 13.3 - Categorization Dual-Group-Consistency Scores Example

- Finally, after the *dual-group-consistency score*s have been calculated they are then used to evaluate how well the machine could categorize a set of entities. In this example, the machine method had exceeded the students score by more than the tolerance, therefor it was able to categorize a set of entities better than the average student.

13.6.3. Dual-Group-Consistency Score Test

The machine categorization results were validated against three student surveys using the machine and student *dual-group-consistency* scores. The *dual-group-consistency* scores were calculated for each of the three surveys and the results recorded in the table below. The machine is considered to have performed the same as the average student if the machine and the class *dual-group-consistency* scores differed by no more than the tolerance value. If the machine's score exceeded the class's score by more than the tolerance value, then the machine is considered to have categorized better than the average student. If the machine's score was less than the class's score by more than the tolerance level, then the machine is considered to have performed worse than the average student in categorizing entities.

A description of the *dual-group-consistency* score table's columns is detailed here. The "Survey" column displays which survey the evaluation results originated from. The "Machine *Dual-Group-Consistency* Score" column displays the categorization method's consistency scores for each of the three surveys. The "Class *Dual-Group-Consistency* Score" column displays the class of student's consistency scores for each of the three surveys. The "Tolerance Value" column displays the tolerance value corresponding to each of the class's *dual-group-consistency* scores.

Survey	**Machine *Dual-Group-Consistency* Score**	**Class *Dual-Group-Consistency* Score**	**Tolerance Value**
Bears	0.76	0.60	0.05
Whales & Dolphins	0.85	0.72	0.06
Spiders	0.80	0.67	0.09

Table 13.4 - Dual-Group-Consistency Score Machine Categorization Evaluation

The criterion for this performance evaluation is the machine method's categorization consistency with the students compared to the categorization consistency among the students themselves. The class consistency score reflects what the average student's *dual-group-consistency* score was. The class's consistency score is the consistency score of each student pairing in the class, excluding any duplicates, averaged together into one score. If the categorization method is said to have performed as well as the average student, this means the method was just as consistent as the student aggregrate was in categorizing a set objects.

The machine categorization method performed better than the average participating student in categorizing a set of objects. The categorization method had a higher *dual-group-consistency* score than the class did in all three surveys. The categorization method can categorize a set of objects better than the average student, since its *dual-group-consistency* score was higher than the class's in all three surveys.

A complete statistical summary of the student surveys and machine categorization output can be found in appendix J.

13.6.4. Investigator's View of the Machine's Categorization Results

The machine categorization method was proficient at categorizing objects, since it did not place any object into a category that it was not a member of. It also induced a few categories that were not specifically mentioned in the natural language text. For example, the machine categorization method created the "toothed dolphin" and "toothless dolphin" categories even though they were not in the text. On the other hand this can lead to some very generic categorizations.

In the experiments there were a few categorizations that were too general. A few examples of this generic categorization were the creation of the "mother bear", "small whale", and "male spider" categories. Another problem recognized in the machine categorization output is the misplacement of

these generic categories in the taxonomy. For example, "mother bear" was placed at the same level in the taxonomy as "sloth bear" and "black bear". Another example of misplacement was placing the "small whale" category at the same level as the "sperm whale" and the "humpback whale". Subsets of species should not be placed at the same level as the sub-species when creating a taxonomy.

Correcting some of the categorization problems experienced in this experiment could be handled in a couple of different ways. One way would be to provide the machine with more natural language text dealing with the subject to be categorized. A second way would be to define the dictionary in more detail, even though this goes against minimal dictionary data definition that was desired for this experiment. Another way would be to create additional relationships out of some of the sub categories and apply them to the sub category's taxonomy level. For example, take "mother bear" and place it underneath the sub species "brown bear" and "black bear" (sub species that were at the same level as "mother bear"), creating the "mother brown bear" and "mother black bear" sub-categories.

13.6.5. Student Survey Inconsistencies

The student surveys showed that not all the students classified animals in the same categories. This diverse categorization displayed by humans has also been noted by:

- Gould (1983) in his studies of biological classification differences between the pheneticists and cladists.
- Dixon (1982) in his studies on noun classification differences between the Ancient Greeks, Indo-Europeans, Romance language speaking peoples, Tshukwe group of Khosans, Bhatnu, Swahili, Dyirbal, and Yidiny.
- Mayr (1984) in his observations of biological classification differences. He detected categorization discrepancies between the schools of numerical phenetics, cladistics, and evolutionary classification.

It seems that cultural differences, environmental factors and previous knowledge affect the way humans look at things and the way they categorize objects. This diverse classification of objects will have a somewhat negative effect upon the machine method categorization results, since those results are uniform by nature. In other words, if several students were asked if an object belonged to a particular category, some would say it would while others would say it did not. Consolidating the student responses into a single number, that number would have a value between 1 or 0 but not equal to 1 or 0, while the machine method gives only 1 or 0 valued responses. This diverse categorization observed in humans can be corroborated by the fact that out of the 75 possible classification possibilities, in the three surveys, the students only responded uniformly to 28 of them.

13.7. Conclusion

Categorization is summarized completely and its importance is made clear with the quotation

> *Man is, by nature, a classifying animal. His continued existence depends, in fact, on his ability to recognize similarities and differences among objects and event in his physical universe and to mark these similarities and differences linguistically.*
> [Berlin, Breedlove, Raven, 1974]

For a machine to show it has conceptualization capabilities it must be able to categorize entities it receives information about with accuracy.

To show that a machine has the conceptualization ability to categorize was a complex process. First, a machine method had to be developed to categorize entities. Second, tools had to be found that could evaluate the results of the categorization method. Next, students had to be surveyed, so there was data

to gauge the accuracy of the machine categorization method. Finally, the results of the evaluations were collected and summarized to determine if a machine with the appropriate method could categorize entities with any degree of accuracy.

Listed below are some of the more prevalent rules that were involved in the machine categorization process.

- Whenever any physical or abstract entity is encountered and can not be found in the semantic net, it becomes a child of the semantic net root and its primary parent is the root.
- An entity's primary parent will always be located at the lowest level possible in the semantic net.
- A subset relation between two entities is determined by the pre-modifiers of the entities and the relation between them.
- When an entity node is labeled with a name and the relation is some form of "be," the entity is put into the net as an instance of another entity.

The above rules only include the "main" logic of the categorization process, since there are an infinite number of ways an entity can be classified dependent upon the content of the input text.

A *dual-group-consistency score* was developed to evaluate the machine categorization method. The score essentially computes the similarity of machine and human categorizations. The machine *dual-group-consistency score* entailed correlating the machine category selections with each student's selections and averaging the coefficients into one score. The class *dual-group-consistency score* involved correlating each student's category selections with every other student's selections, excluding duplicated pairings, and averaging the coefficients into one score. A tolerance value was also calculated by taking the variance of the class's correlation coefficients. The extent that a machine can categorize a set of entities was measured by comparing the machine's *dual-group-consistency score* with the class's *dual-group-consistency score*. The scores were considered the same if they differed by no more than the tolerance value.

Three surveys were taken from a college student population to evaluate the performance of the machine's categorization methods. In each survey the students were handed a packet that consisted of an article and a set of questions. The article received by the students was about specific entities. The students were to read the article and use it as a reference in answering five questions. The five questions in each survey the students were given had the following format, "*Circle the letter of the following categories that include Entity X*". The data derived from the student responses to the survey was then used to validate the results of the machine's categorization method.

The extent that a machine could categorize a set of objects was computed by comparing its *dual-group-consistency score* with the *dual-group-consistency score* of a class of students. In this experiment the categorization method performed better than the average student in all three surveys. This means that the machine categorized a set of objects with more similarity than the average student when both their category selections were compared with the class's selections. The machine in this experiment could categorize a set of related entities to the extent of being better than the average student.

It also can be verified by looking at the category selections in appendix I, that the machine always classified an entity the same as the majority of the students did in all three surveys. The aforementioned measures indicate that a machine can categorize a set of objects with accuracy, since its category selections are similar to human category selections.

14. Entity Importance

14.1. Introduction

A critical tool in the conceptualization process is to determine among a group of concepts the most important concept. This tool is essential to our survival and is required in day to day living. It also helps us choose what foods to eat, what careers to take, where to live, whom to make friends with, what house to buy, what materials to build with, etc. If humans lacked this valuable mechanism, they probably wouldn't exist today. For the aforementioned reasons, determining entity importance is essential to conceptualization. If a machine has the ability to determine entity importance then a good and quantitative measure of a machine's conceptualizing facilities can be determined by how well it can rank a group of entities by importance.

14.2. Purpose

The goal of this chapter is to show to what extent a machine can rank entities by importance. This is done by demonstrating the degree of similarity between machine and human importance rankings. The demonstration includes:

- Illustrating and reviewing the process involved in entity importance ranking
- Explaining the machine method's importance output
- Discussing the student survey importance questions
- Explicating the measures used to evaluate machine performance
- Validating machine output using the student survey results

With the previous steps executed, the machine importance ranking method can be evaluated to some degree.

A secondary goal of this experiment was to determine which machine method out of five, performed the best in ranking entities by importance. This was done by selecting the machine method with the best performance record. Performance was based upon on how similar a machine method was to humans in ranking entities by importance.

14.3. Process

The process of ranking related entities by importance within a semantic net was a complicated task that was performed in three steps. The first step tried five different methods to calibrate each entity's significance within the net. The second step gathered a subset of entities to be ranked. The third step used an algorithm to rank the subset of entities in descending order according to their importance scores. The five different scoring methods, subset selection process and ranking algorithm are illustrated and reviewed in detail in the following sections.

14.3.1. Five Scoring Methods

The five scoring methods evaluated to determine which one simulated human importance ranking the best were:

- *Emanating Arcs Method*
- *Entering Arcs Method*
- *Arcs In & Out Method*
- *All Descendants Method*
- *All Ancestors Method*

The different methods were used to decide what entity characteristics and attributes contributed the most in predicting human importance rankings. Each method's description, algorithm and sample calculation (*using the semantic net pictured below*) are detailed below.

Figure 14.1 - Sample Semantic Net Used for Importance Calculations

14.3.1.1. Emanating Arcs Method

The *Emanating Arcs Method* is a simple method that calculates an entity's importance score by counting the number of arcs that emanate from the entity. This is done for all the entities that are to be ranked in a particular subset. The arcs are counted during the formation of the semantic net. Every time a new relation is added to an entity and the entity is the subject of that relation, the entity's emanating arc counter is incremented by 1. The entity emanating arc counter is decremented by 1 every time a relation is removed from the entity where the entity is the subject of that relation. The value of the emanating arc counter is used for the importance score when the *Emanating Arcs Method* is selected.

An example of how the *Emanating Arcs Method* is used to calculate an importance score for several entities is demonstrated using figure 14.1 from above. Displayed below are two entities, toothed whale and orca whale, with their corresponding importance scores and how they were derived.

- The toothed whale has an importance score of 3, since it has the following three relations that it is the subject of:
  - ***toothed whale*** *SUBSET OF whale*
  - ***toothed whale*** *HAVE teeth*
  - ***toothed whale*** *INCLUDES sperm whale*
- The orca whale has an importance score of 5, since it has the following five relations that it is the subject of:
  - ***orca whale*** *ISA whale*
  - ***orca whale*** *HAVE sharp teeth*
  - ***orca whale*** *IS smart*
  - ***orca whale*** *kill dolphin*
  - ***orca whale*** *HAVE black fins*

The *Emanating Arcs Method* is easy to calculate, since it only takes into account the particular entity that the importance score is being calculated for.

14.3.1.2. Entering Arcs Method

The *Entering Arcs Method* is similar to the *Emanating Arcs Method*. This method calculates an entity's importance score by counting the number of arcs that enter into an entity. This is done for all the

entities that are to be ranked in a particular subset. The arcs are counted during the formation and manipulation of the semantic net. Every time a new relation is added to an entity and the entity is the object of that relation, the entity's entering arc counter is incremented by 1. The entity entering arc counter is decremented by 1 every time a relation is removed from the entity, where the entity is the object of that relation. The value of the entering arc counter is used for the importance score when the *Entering Arcs Method* is selected.

An example of how the *Entering Arcs Method* is used to calculate an importance score for several entities is demonstrated using figure 14.1 from above. Displayed below are two entities, toothed whale and orca whale, with their corresponding importance scores and how they were derived.

- The toothed whale has an importance score of 2, since it has the following two relations that it is the object of:
  - *whale HAS SUBSET **toothed whale***
  - *sperm whale ISA **toothed whale***
- The dolphin has an importance score of 1, since it has the following relation that it is the object of:
  - *whale INCLUDES **orca whale***

The *Entering Arcs Method* is also easy to calculate, since it only takes into account the particular entity that the importance score is being calculated for.

14.3.1.3. Arcs In & Out Method

The *Arcs In & Out Method* is a combination of the *Emanating Arcs Method* and the *Entering Arcs Method*. The *Arcs In & Out Method* calculates the importance score for each entity in the subset to be ranked, by adding the number of arcs emanating from an entity with the number of arcs entering into an entity. For example, if an entity had 3 arcs emanating from it and 5 arcs entering into it, then its importance score would be 8. The emanating and entering arcs are counted during the formation and manipulation of the semantic net.

An example of an importance score calculated using the *Arcs In & Out Method* is demonstrated below using figure 14.1. Displayed below are two entities, toothed whale and orca whale, with their corresponding importance scores and how they were derived.

- The toothed whale has an importance score of 5, since it has the following five relations that it is the subject and object of:
  - ***toothed whale** SUBSET OF whale*
  - ***toothed whale** HAVE teeth*
  - ***toothed whale** INCLUDES sperm whale*
  - *whale HAS SUBSET **toothed whale***
  - *sperm whale ISA **toothed whale***
- The orca whale has an importance score of 6, since it has the following six relations that it is the subject and object of:
  - ***orca whale** ISA whale*
  - ***orca whale** HAVE sharp teeth*
  - ***orca whale** IS smart*
  - ***orca whale** kill dolphin*

- ***orca whale*** *HAVE black fins*
- *whale INCLUDES* ***orca whale***

14.3.1.4. All Descendants Method

The *All Descendants Method* calculates an entity's importance score by counting the number of arcs that emanate from the entity plus the accumulation of the number of arcs that emanate from its primary descendants. This is done for all the entities contained in the semantic net, after it has been constructed and manipulated. When this algorithm is used, entities at the bottom of the net will have lower scores than those of their ancestors near the top. The *All Descendants Method* favors those entities with lots of primary descendants. Displayed below is the algorithm for the *All Descendants Method*.

The algorithm starts at the root of the net and is recursively passed down to each of its children:

- The entity's importance score is set to zero.
- The entity passes the procedure down to all of its children.
- The process is continued until the bottom of the net is reached.
- The entity sets its importance score equal to the number of arcs that emanate from it.
- The entity then increments its importance score by the number of arcs that emanate from each of its children's entities.

Algorithm 14.1a - All Descendants Method (summation)

```
imp_score  /*Importance score*/
entity     /*Current entity*/
relation[] /*Arcs emanating from the current entity*/
children[] /*Current entity's children connected by relation*/

/*AllDescendantMethod Initialization*/
entity = root of the semantic net
entity.AllDescendantMethod

/*Recursively Call The AllDescendantMethod Algorithm*/
entity.imp_score = 0
i = 0
While i < entity.relation_quantity Do
    entity.imp_score = entity.imp_score +
                       relation[i].relation_quantity
    If relation[i] = ("INCLUDES" Or "HAS SUBSET" Or "INSTANCE") Then
        entity.children[i].AllDescendantMethod  /*Primary Children*/
    End If
    i = i + 1
End While

/*Accumulate The Descendants Importance Scores*/
i = 0
While i < entity.relation_quantity Do
    If relation[i] = ("INCLUDES" Or "HAS SUBSET" Or "INSTANCE") Then
        entity.imp_score = entity.imp_score + children[i].imp_score
    End If
    i = i + 1
```

```
    End While
```

Algorithm 14.1b - All Descendants Method (technical)

The *All Descendants Method* algorithm is called recursively for each of the entity's primary children starting at the root of the semantic net. As the algorithm traverses down the net it counts the number of arcs that emanate from the calling entity. This process is continued until the bottom of the net is reached. When the algorithm reaches the net's bottom, the importance score is then computed as being the number of arcs emanating from the bottom entity. The bottom entity then passes up its importance score to its primary parent. The primary parent then calculates its importance score by adding the number of its emanating arcs along with its primary children's importance scores. After the parent's importance score has been calculated, the score is then passed on up to its primary parent. This process is continued until the root of the semantic net is reached. At that time, all the entities within the net will have had their importance scores calculated.

An example of the *All Descendants Method* calculation is demonstrated below using figure 14.1. Displayed below are two entities, sperm whale and toothed whale, with their corresponding importance scores and how they were derived.

- The sperm whale has an importance score of 2, since it has the following two relations that it is the subject of:
  - ***sperm whale*** *ISA toothed whale*
  - ***sperm whale*** *grows 65 feet*
- The toothed whale has an importance score of 5, since it has the following three relations that it is the subject of plus the importance score of 2 from its descendant (*primary child*), the sperm whale:
  - ***toothed whale*** *SUBSET OF whale*
  - ***toothed whale*** *HAVE teeth*
  - ***toothed whale*** *INCLUDES sperm whale*

14.3.1.5. All Ancestors Method

The *All Ancestors Method* has the reverse logic of the *All Descendants Method*. It calculates an entity's importance score by counting the number of arcs that emanate from the entity, plus the accumulation of the number of arcs that emanate from its primary ancestors. This is done for all the entities contained in the net, after it has been constructed and manipulated. When this algorithm is used, entities at the bottom of the net will have higher importance scores than those of their ancestors near the top. Displayed below is the *All Ancestors Method* algorithm.

The algorithm starts at the root of the net and is recursively passed down to each of its children:

- Set importance score equal to the number of arcs that emanate from the entity. This excludes entities less than a distance of 1 from the net's root.
- If the entity is a distance of 2 or more from the root of the net and not an "instance" then the entity a its primary parent's importance score to its score.
- The entity then initiates the algorithm among its primary children.
- The process is continued until the bottom of the net is reached.

Algorithm 14.2a - All Ancestors Method (summation)

```
imp_score  /*Importance score*/
entity     /*Current entity*/
relation[] /*Arcs emanating from the current entity*/
children[] /*Current entity's children connected by relation*/
distance   /*Distance from the root of the semantic net*/
subset_relations  /*Number of "HAS SUBSET" relations*/

/*AllAncestorMethod Initialization*/
entity = root of the semantic net
entity.AllAncestorMethod

/*AllAncestorMethod Algorithm*/

/*Count The Number Of Emanating Arcs*/
i = 0
entity.imp_score = 0
While i < entity.relation_quantity Do
    entity.imp_score = entity.imp_score + relation[i].relation_quantity
    i = i + 1
End While

/*Accumulate The Ancestor's Importance Scores*/
If entity.distance > 1 Then /*Excludes The Roots Emanating Arcs*/
    i = 0
    While i < entity.relation_quantity Do
        If relation[i] = ("ISA" Or "SUBSET OF") Then /*Ancestors*/
            entity.imp_score  = entity.imp_score + children[i].imp_score
                              - children[i].subset_relations
        End If
        i = i + 1
    End While
End If

/*Recursively Call The AllAncestorMethod Algorithm*/
i = 0
While i < entity.relation_quantity Do
    If relation[i] = ("INCLUDES" Or "HAS SUBSET" Or "INSTANCE") Then
        entity.children[i].AllAncestorMethod    /*Primary Children*/
    End If
    i = i + 1
End While
```

Algorithm 14.2b - All Ancestors Method (technical)

The *All Ancestors Method* works in the following way, less some minor details that are explained in detail in the next paragraph. The method starts from the root of the net and sets each entity's importance score to the number of arcs that emanate from the entity. Next, if the entity has a distance of two or more from the root and is not an entity instance, then it adds its primary parent's importance score to its importance score. When it is finished accumulating its primary parent's importance score, the entity initiates the method among its primary children. This process is continued until the bottom of the net is reached.

The *All Ancestors Method* is more complex than the *All Descendants Method* for a couple of reasons. First, the *All Ancestors Method* has logic to exclude the root of the semantic net, first level entities and entity instances from accumulating their ancestor's importance score. This is done, since the root entity's emanating arcs are excluded from the entity's importance score. Ancestor scores were

excluded from the entity instances, since they are deemed less importance than their parent entities. The second thing that is different from the descendant's method is that the "HAS SUBSET" relations were excluded from the primary ancestor's importance scores. This was done to place more importance upon entities with the lots of subsets but not upon their children.

An example of the *All Ancestors Method* calculation is demonstrated below using figure 14.1 for input. Displayed below are three entities, mammal, whale and toothed whale, with their corresponding importance scores and how they were derived.

- The mammal has an importance score of 4, since it has the following four relations that it is the subject of:
  - ***mammal*** *ISA entity*
  - ***mammal*** *breathe air*
  - ***mammal*** *INCLUDES whale*
  - ***mammal*** *INCLUDES dolphin*
- The whale has an importance score of 10, since it has the following six relations that it is the subject of plus the importance score of 4 from its primary parent, the mammal entity:
  - ***whale*** *ISA mammal*
  - ***whale*** *IS big*
  - ***whale*** *HAS SUBSET toothed whale*
  - ***whale*** *INCLUDES orca whale*
  - ***whale*** *INCLUDES blue whale*
  - ***whale*** *HAVE fins*
- The toothed whale has an importance score of 12, since it has the following three relations that it is the subject of plus the importance score of 10 from its primary parent the whale entity, and minus 1 for its "HAS SUBSET" relation:
  - ***toothed whale*** *SUBSET OF whale*
  - ***toothed whale*** *HAVE teeth*
  - ***toothed whale*** *INCLUDES sperm whale*

14.3.2. Subset Selection Process

The subset selection process creates an array of entities to be ranked by performing the following tasks. First the process inquires the user for what subset of entities he or she would like to be ranked. Next, the user inputs an entity that is contained within the semantic net. The process then finds the user specified entity in the net and gathers all of its primary descendants. Two algorithms are involved in collecting the subset of entities to be ranked and are detailed below. The first algorithm, *Find Entity*, finds the user selected entity in the net. The second algorithm, referred to as *Collect Entities*, gathers the primary descendants of the entity returned by the *Find Entity* algorithm and places them into an array.

The *Find Entity* algorithm is a recursive procedure. It starts at the root of the semantic net and works itself down into the net following the paths leading to each entity's primary children. If none of the primary children match the user requested entity, then each of the children call the *Find Entity* algorithm to match their children against the requested entity. The algorithm doesn't stop until the user

requested entity is found or the bottom of the net is reached. Displayed below is the algorithm used in this experiment.

```
entity     /*Current entity*/
relation[] /*Arcs emanating from the current entity*/
children[] /*Current entity's children connected by relation*/
relation_quantity  /*Number of arcs that emanate from the entity*/
requested_entity   /*User requested entity*/

/*FindEntity Initialization*/
entity = root of the semantic net
entity.FindEntity(requested_entity)

/*FindEntity Algorithm*/

/*Is the current entity the user request entity*/
If (entity = requested_entity And entity NOT instance) Then
    Return entity
End If

/*Recursively Call The FindEntity Algorithm*/
i = 0
While i < entity.relation_quantity Do
    If relation[i] = ("INCLUDES" Or "HAS SUBSET" Or "INSTANCE") Then
        entity.children[i].FindEntity(requested_entity)
    End If
    i = i + 1
End While
```

Algorithm 14.3 - Find Entity

The *Collect Entities* algorithm is a little more complex than the *Find Entity* algorithm, since it has to gather a group of entities into an array. The *Collect Entities* algorithm is recursive and starts from the user requested entity returned by the *Find Entity* algorithm. The *Collect Entities* method puts all the primary children of the requested entity into an array, excluding children that are instances. Then each of the children that were put into the array, initiate the *Collect Entities* algorithm to add their primary children to the array, excluding their entity instances. This process is continued until all the primary descendants of the requested entity are collected into an array. The algorithm is displayed below.

```
entity     /*Current entity*/
relation[] /*Arcs emanating from the current entity*/
children[] /*Current entity's children connected by relation*/
relation_quantity  /*Number of arcs that emanate from the entity*/
requested_entity   /*User requested entity*/
collection[]       /*The subset of entities to be gathered*/
cllctn_qty         /*The number of entities gathered*/

/*CollectEntities Initialization*/
entity = requested_entity /*returned by the FindEntity algorithm*/
cllctn_qty = 0
entity.CollectEntities(collection,cllctn_qty)

/*CollectEntities Algorithm*/

/*Recursively Call CollectEntities Algorithm and gather entities*/
i = 0
While i < entity.relation_quantity Do
```

```
        If relation[i] = ("INCLUDES" Or "HAS SUBSET") Then
            collection[cllctn_qty] = entity.children[i]
            cllctn_qty = cllctn_qty + 1
            entity.children[i].CollectEntities(collection,cllctn_qty)
        End If
        i = i + 1
    End While
```

Algorithm 14.4 - Collect Entities

14.3.3. Ranking Algorithm

The ranking algorithm consists of two processes, one that sort the entities and the other that assigns a rank to each of the entities. The sorting process pulls in an array of entities and sorts them in descending order. The sorting process is detailed below.

```
    entities[]     /*the subset of entities to be ranked*/
    temp_entity    /*temporary entity place holder*/
    entity_qty     /*the number of entities contained in entities[]*/
    imp_score      /*entity importance score previously calculated*/

    /*Sorting Process*/
    i = 0
    While i < entity_qty Do
        j = i + 1
        While j < entity_qty Do
            If entities[i].imp_score < entities[j].imp_score Then
                temp_entity = entities[i]
                entities[i] = entities[j]
                entities[j] = temp_entity
            End If
            j = j + 1
        End While j
        i = i + 1
    End While i
```

Algorithm 14.5 - Sorting Process

The second process of the ranking algorithm assigns a rank to each of the entities in the array produced by the *Collect Entities* algorithm. The entity at the beginning of the array receives a rank of 1. The second entity receives a rank of two, unless it has an importance score equal to the previous entity in the array, then it would receive the same rank as the preceding entity. The next entity would receive a rank of 3 unless it had an importance score equal to the previous entity, then it would receive the previous entity's rank. This ranking scheme continues until all the entities in the array have been ranked. Entities with the same importance scores receive the same rank. The ranking process is illustrated below.

```
    entities[]     /*the subset of entities to be ranked*/
    entity_qty     /*the number of entities contained in entities[]*/
    imp_score      /*entity importance score previously calculated*/

    /*Ranking Process*/
    i = 0
    j = 1
    k = 1
    last = 1
    While i < entity_qty Do
```

```
        If i > 0 And entities[i-1].imp_score Not = entities[i].imp_score      Then
            j = j + 1
        End If
        If j Not = last Then j = k
        entities[i].rank = j
        k = k + 1
        last = j
        i = i + 1
    End While i
```

Algorithm 14.6 - Ranking Process

The ranking algorithm can be summarized by the following two steps:

- *Sort an array of related entities in descending order using a bubble sort.*
- *Assign each entity's rank equal to its position in the array unless its importance score is equal to the previous entity's score, then assign it the rank equal to that of the previous entity's rank.*

For example, given an array of five bears, grizzly bear, black bear, brown bear, kodiak bear, and polar bear, assigned with the importance scores of 30, 40, 10, 50 and 30 respectively. The bears are then sorted in descending order resulting in the following sequence: kodiak bear, black bear, grizzly bear, polar bear, and brown bear. Finally the bears are given the ranks of 1, 2, 3, 3, and 5 respectively from the previous sentence. The kodiak bear received a rank of 1, the black bear a rank of 2, and etc.

14.4. Machine Output

The machine displays the result of the entity importance ranking by showing an array of ranked entities in a vertical column. The entities are written in descending order according to their importance score while the rank of each entity is shown to its right. An example of an entity importance ranking result is displayed below.

```
whale#0 Most Important Ranking
    orca whale#0 1
    sperm whale#0 2
    toothed#0 whale#0 3
    blue whale#0 3
    toothless#0 whale#0 5
    humpback whale#0 5
    largest#0 toothed#0 whale#0 7
    killer whale#0 8
    small#0 whale#0 8
```

Example 14.1 - Sample Importance Ranking Output

The importance ranking for each article is displayed in the following sub-sections. Each sub-section will be headed by the article used for input, followed by the corresponding ranking from each machine method.

14.4.1. Bear Importance Ranking Machine Output

```
bear#0 Most Important Ranking
   black bear#0 1
   polar bear#0 2
   brown bear#0 3
   sun bear#0 4
```

```
   mother#1 bear#0 5
   spectacled bear#0 6
   mother#1 sloth bear#0 6
   sloth bear#0 6
   fiercest#0 brown bear#0 6
   smallest#0 bear#0 6
   biggest#0 brown bear#0 6
   kodiak bear#0 12
   grizzly bear#0 12
   blue bear#0 12
   moon bear#0 12
```

Figure 14.2 - Emanating Arcs Method Importance Ranking (Bear Article)

```
bear#0 Most Important Ranking
   brown bear#0 1
   polar bear#0 2
   sun bear#0 3
   kodiak bear#0 3
   moon bear#0 5
   black bear#0 5
   sloth bear#0 5
   grizzly bear#0 5
   smallest#0 bear#0 5
   biggest#0 brown bear#0 5
   blue bear#0 11
   spectacled bear#0 11
   fiercest#0 brown bear#0 11
   mother#1 bear#0 14
   mother#1 sloth bear#0 14
```

Figure 14.3 - Entering Arcs Method Importance Ranking (Bear Article)

```
bear#0 Most Important Ranking
   black bear#0 1
   brown bear#0 2
   polar bear#0 2
   sun bear#0 4
   sloth bear#0 5
   kodiak bear#0 5
   smallest#0 bear#0 5
   biggest#0 brown bear#0 5
   spectacled bear#0 9
   grizzly bear#0 9
   moon bear#0 9
   fiercest#0 brown bear#0 9
   mother#1 bear#0 9
   blue bear#0 14
   mother#1 sloth bear#0 14
```

Figure 14.4 - Arcs In & Out Method Importance Ranking (Bear Article)

```
bear#0 Most Important Ranking
   brown bear#0 1
   black bear#0 2
   polar bear#0 3
   smallest#0 bear#0 4
   sloth bear#0 5
   sun bear#0 5
```

```
   fiercest#0 brown bear#0 7
   mother#1 bear#0 7
   biggest#0 brown bear#0 7
   spectacled bear#0 10
   mother#1 sloth bear#0 10
   kodiak bear#0 12
   grizzly bear#0 12
   blue bear#0 12
   moon bear#0 12
```

Figure 14.5 - All Descendants Method Importance Ranking (Bear Article)

```
bear#0 Most Important Ranking
   moon bear#0 1
   black bear#0 2
   polar bear#0 3
   brown bear#0 4
   kodiak bear#0 5
   grizzly bear#0 5
   biggest#0 brown bear#0 7
   fiercest#0 brown bear#0 7
   sun bear#0 9
   mother#1 bear#0 10
   mother#1 sloth bear#0 10
   spectacled bear#0 12
   smallest#0 bear#0 12
   sloth bear#0 12
   blue bear#0 15
```

Figure 14.6 - All Ancestors Method Importance Ranking (Bear Article)

14.4.2. Whale Importance Ranking Machine Output

```
whale#0 Most Important Ranking
   orca whale#0 1
   sperm whale#0 2
   toothed#0 whale#0 3
   blue whale#0 3
   toothless#0 whale#0 5
   humpback whale#0 5
   largest#0 toothed#0 whale#0 7
   killer whale#0 8
   small#0 whale#0 8
```

Figure 14.7 - Emanating Arcs Method Importance Ranking (Whale Article)

```
whale#0 Most Important Ranking
   sperm whale#0 1
   orca whale#0 1
   killer whale#0 3
   largest#0 toothed#0 whale#0 4
   toothed#0 whale#0 4
   blue whale#0 4
   humpback whale#0 4
   small#0 whale#0 4
   toothless#0 whale#0 9
```

Figure 14.8 - Entering Arcs Method Importance Ranking (Whale Article)

```
whale#0 Most Important Ranking
   orca whale#0 1
   sperm whale#0 2
   toothed#0 whale#0 3
   blue whale#0 3
   humpback whale#0 5
   toothless#0 whale#0 6
   largest#0 toothed#0 whale#0 6
   killer whale#0 6
   small#0 whale#0 9
```

Figure 14.9 - Arcs In & Out Method Importance Ranking (Whale Article)

```
whale#0 Most Important Ranking
   toothed#0 whale#0 1
   orca whale#0 2
   largest#0 toothed#0 whale#0 3
   sperm whale#0 4
   blue whale#0 4
   toothless#0 whale#0 6
   humpback whale#0 6
   killer whale#0 8
   small#0 whale#0 8
```

Figure 14.10 - All Descendants Method Importance Ranking (Whale Article)

```
whale#0 Most Important Ranking
   killer whale#0 1
   orca whale#0 2
   toothed#0 whale#0 3
   blue whale#0 3
   toothless#0 whale#0 5
   humpback whale#0 5
   small#0 whale#0 7
   sperm whale#0 8
   largest#0 toothed#0 whale#0 9
```

Figure 14.11 - All Ancestors Method Importance Ranking (Whale Article)

14.4.3. Spider Importance Ranking Machine Output

```
spider#0 Most Important Ranking
   garden spider#0 1
   wolf spider#0 2
   crab spider#0 3
   water spider#0 4
   mother#1 spider#0 5
   zebra spider#0 5
   malmignette#0 5
   trapdoor spider#0 8
   black widow#0 8
   funnel-web#0 10
   house spider#0 10
   hammock spider#0 10
   little#0 zebra spider#0 10
   male#0 spider#0 10
   jumping spider#0 10
   female#1 malmignette#0 10
```

```
   American#0 black widow#0 17
   bird-eating spider#0 17
```

Figure 14.12 - Emanating Arcs Method Importance Ranking (Spider Article)

```
spider#0 Most Important Ranking
   zebra spider#0 1
   jumping spider#0 2
   wolf spider#0 3
   crab spider#0 3
   trapdoor spider#0 5
   water spider#0 5
   malmignette#0 5
   black widow#0 5
   garden spider#0 5
   hammock spider#0 10
   house spider#0 10
   funnel-web#0 10
   American#0 black widow#0 10
   bird-eating spider#0 10
   male#0 spider#0 15
   mother#1 spider#0 15
   female#1 malmignette#0 15
   little#0 zebra spider#0 15
```

Figure 14.13 - Entering Arcs Method Importance Ranking (Spider Article)

```
spider#0 Most Important Ranking
   garden spider#0 1
   wolf spider#0 2
   crab spider#0 3
   zebra spider#0 4
   water spider#0 5
   jumping spider#0 6
   malmignette#0 6
   black widow#0 8
   trapdoor spider#0 8
   mother#1 spider#0 10
   funnel-web#0 11
   house spider#0 11
   hammock spider#0 11
   little#0 zebra spider#0 14
   female#1 malmignette#0 14
   male#0 spider#0 14
   American#0 black widow#0 14
   bird-eating spider#0 14
```

Figure 14.14 - Arcs In & Out Method Importance Ranking (Spider Article)

```
spider#0 Most Important Ranking
   garden spider#0 1
   wolf spider#0 2
   crab spider#0 2
   jumping spider#0 4
   water spider#0 4
   zebra spider#0 6
   trapdoor spider#0 6
   malmignette#0 6
   black widow#0 9
```

```
   mother#1 spider#0 9
   house spider#0 11
   hammock spider#0 11
   little#0 zebra spider#0 11
   male#0 spider#0 11
   female#1 malmignette#0 11
   funnel-web#0 11
   American#0 black widow#0 17
   bird-eating spider#0 17
```

Figure 14.15 - All Descendants Method Importance Ranking (Spider Article)

```
spider#0 Most Important Ranking
   garden spider#0 1
   wolf spider#0 2
   little#0 zebra spider#0 3
   crab spider#0 3
   funnel-web#0 5
   zebra spider#0 5
   water spider#0 5
   female#1 malmignette#0 8
   mother#1 spider#0 9
   trapdoor spider#0 9
   malmignette#0 9
   American#0 black widow#0 12
   black widow#0 12
   house spider#0 14
   hammock spider#0 14
   jumping spider#0 14
   male#0 spider#0 14
   bird-eating spider#0 18
```

Figure 14.16 - All Ancestors Method Importance Ranking (Spider Article)

14.5. Entity Importance Student Survey

The medium used to validate the machine importance process was a survey given to a class of college students. In the survey two questions were asked to calibrate the student's importance ranking of seven entities after they had read a children's book on a particular animal category. At first, only question 3, "*rank the entities by their importance*", was going to be asked, but the concept of *Most Important* entity can be subjective, so question 1, "*rank the entities by the amount of knowledge you have learned about them*", was added to come up with another gauge of importance. They were both asked in an attempt to see which question would calibrate the student's importance ranking the best. Also included with the two questions was a definition of the term, "*Most Important*", that was located in the front of the survey. The format of the two questions are displayed below.

1) Rank the following objects according to what you learned from the article. Rank them in descending order. Give the object you are most knowledgeable about a "1" and the one you know <u>least</u> about a rank of "7". All objects must be ranked.

___ entity A
___ entity B
___ entity C
___ entity D
___ entity E
___ entity F

___ *entity G*

Example 14.2 - Most Knowledgeable Question Format

3) Rank the following objects according to what is the ***Most Important*** *object from the previous article. Rank them in descending order. Give the Most Important object a "1" and the <u>least</u> important a rank of "7". All objects must be ranked.*

___ *entity A*
___ *entity B*
___ *entity C*
___ *entity D*
___ *entity E*
___ *entity F*
___ *entity G*

Example 14.3 - Most Important Question Format

The numerical result of the importance survey was a two by seven matrix for each student that answered the survey. Column one of the matrix captured the student rankings for entity knowledge and column two captured the student rankings for entity importance. The entries in the matrix had values between one and seven representing the student ranking responses. For example, a student answering question 1 with the following responses: entity A = rank of 3, entity B = rank of 1, entity C = 6, D = 4, E = 2, F = 7, and G = 5; question 3 with the responses of: A = 2, B = 1, C = 7, D = 6, E = 4, F = 3, and G = 5; would be represented by the following matrix.

	Question 1 (knowledge)	*Question 3 (importance)*
Entity A	**3**	**2**
Entity B	**1**	**1**
Entity C	**6**	**7**
Entity D	**4**	**6**
Entity E	**2**	**4**
Entity F	**7**	**3**
Entity G	**5**	**5**

Example 14.4 - Student Survey Most Importance Response Matrix

14.6. Validation of Machine Output

In the absence of a standardized testing instrument, measuring expert performance is difficult. Classically, human experts measure whether some new person is also an expert by how consistent the new person is with the recognized experts. A new person can be thought of as an expert if his performance is consistent with expert performance to the degree that the experts' performances are consistent with each other.

Two consistency scores were developed to evaluate the five machine importance methods.

- *Rank-Group-Consistency* score - Measures to what extent a machine method can rank a set of objects by importance.
- *Rank-Individual-Consistency* score - Determines which machine method performs the best at ranking a set of objects by importance.

14.6.1. Extent of Machine Importance Ranking

The following steps were used to determine to what extent a machine can rank a set of objects by importance:

- The *rank-group-consistency* score was defined.
- An explanation of how the *rank-group-consistency* score is computed and used was given.
- The *rank-group-consistency* score test was performed. This was done to determine the extent a machine can rank set of objects by importance.

14.6.1.1. Rank-Group-Consistency Score Defined

The objective of the *rank-group-consistency score* is to measure to what extent a machine method can rank a set of objects by importance. Determining this measure is done by comparing the machine's *rank-group-consistency score* to the *rank-group-consistency score* of the class. The *rank-group-consistency score* comparison is used to determine if the computer's rankings were similar to a class of students' rankings. This was done by comparing the computer's rankings to each student in the class and comparing each student's rankings with every other student's rankings in the class. The machine method's and class's consistency scores are then compared. This determines if the machine method performed below, the same, or higher than the average student in ranking a set of objects by importance.

A *rank-group-consistency score* was developed for the machine. This was formulated by, comparing the machine rankings to each student's rankings in the class. This resulted in an array of similarity scores. The similarity scores were then aggregated into one *rank-group-consistency score* by averaging them together. The idea is to discover how consistent the machine's rankings were with the rankings of the student aggregate. The *rank-group-consistency score* indicates how similar the machine was to the student aggregate in ranking a set of objects. A high average indicates a high level similarity.

A *rank-group-consistency score* was developed for the students as a group. This was done by first, comparing each student's rankings with every other student's rankings, excluding duplicate pairings. The similarity scores for the students were then aggregated into one *rank-group-consistency score* by averaging them together. The idea being to discover how generally consistent the students were in their rankings amongst themselves.

The extent that a machine can rank a set of objects by importance can be measured by comparing the machine's *rank-group-consistency score* with the class's *rank-group-consistency score*. The scores were considered the same if they differed by no more than a tolerance value. In this experiment, the tolerance value was the variance of the class's *rank-group-consistency score*.

The similarity score between two sets of rankings was measured by calculating the Spearman Correlation Coefficient of the two. A correlation coefficient with a value close to one indicates a high level of similarity between the two. A value near zero indicates little or no similarity between the two. A value close to negative one indicates rankings that go in opposite directions.

14.6.1.2. Spearman's Rank Correlation Coefficient

Spearman's rank correlation coefficient tests the correlation between two ranked variables. It was used in this experiment to indicate if a pair of entity rankings, given by two students or a student and machine method, were similar. The null hypothesis states that there is no association between the two ranked variables while the alternative hypothesis says that there is. Spearman's rank correlation

denotes a near perfect straight line fit between two rankings with a correlation value close to 1, while a correlation value near 0 indicates little to no relationship between the two rankings. The formula used to calculate Spearman's rank correlation coefficient is detailed below.

where

x_i = Represents the i^{th} ranked entity from the first set of rankings.

y_i = Represents the i^{th} ranked entity from the second set of rankings.

n = The number of entities in a ranked set.

Rejection Region: Reject Ho, if S ≥ 0.714 (critical value from Spearman's rank correlation coefficient table with n = 7 and α = 0.05 "95% confidence").

Ho: There is no association between the ranked sets.

Ha: There is an association between the ranked sets.

[Mendenhall, Wackerly, Scheaffer, 1990]

The two following examples demonstrate how the Spearman test indicates if there is an association between two ranked sets. The first example presents two sets that have no association between them and the second example shows two sets that are associated.

	Survey Rankings						
Student	**hump-back whale**	**toothed whale**	**toothless whale**	**blue whale**	**orca whale**	**small whale**	**sperm whale**
1	1	2	3	4	5	6	7
2	4	5	6	7	1	2	3

= 98

= = 28

= = 140

Since S < 0.714, the null hypothesis is accepted and the alternate hypothesis is rejected. i.e. There is no association between the two student whale rankings with 95% confidence. S having a value of -0.5 implies that the two student whale rankings are negatively correlated and have little in common.

Example 14.5 - Spearman's Rank Correlation (no association between sets)

Looking at the above rankings, it can be seen that neither student gave the same rank for any of the whales and that the ranges between each pair of rankings were fairly high. When these two symptoms occur, the correlation between the two ranked sets will always be low.

	Survey Rankings						
Student	**hump-back whale**	**toothed whale**	**toothless whale**	**blue whale**	**orca whale**	**small whale**	**sperm whale**

1	1	2	3	4	5	6	7
2	2	1	3	6	4	5	7

= 136
= = 28
= = 140

Since S > 0.714 The null hypothesis is rejected and the alternate hypothesis is accepted. i.e. There is an association between the two student whale rankings with 95% confidence. S having a value of 0.851 implies that the two student whale rankings are correlated.

Example 14.6 - Spearman's Rank Correlation (associated sets)

Observing the student rankings, it can be seen that the two students gave similar rank values for each pair of whales. When the ranks differed, the range between the pair was small. When these two symptoms occur, the correlation between the two ranked sets will always be high.

14.6.1.3. Rank-Group-Consistency Score Example

To show how a machine method's and student's *rank-group-consistency score*s were calculated and used in the importance evaluation, an example is detailed below.

- First, five students are given an article about bears to read.
- Second, each student is asked to answer the following question.

  1) Rank the following objects according to what is the Most Important *object from the article. Rank them in descending order. Give the most important object a "1" and the <u>least</u> important object a rank of "7". All objects must be ranked.*

  ___ *polar bear*
  ___ *brown bear*
  ___ *sun bear*
  ___ *kodiak bear*
  ___ *sloth bear*
  ___ *grizzly bear*
  ___ *black bear*

- The machine method also reads the same article about bears and answers the above question.
- The five students and the machine answers to the above question are numerated in the table below.

Students/Machine	**s1**	**s2**	**s3**	**s4**	**s5**	**machine**
polar bear	2	1	3	2	2	2
brown bear	3	3	1	3	3	2
sun bear	4	7	7	6	4	4
kodiak bear	5	4	5	7	6	5
sloth bear	6	6	6	4	5	5
grizzly bear	7	5	4	5	7	7
black bear	1	2	2	1	1	1

Table 14.1 - Sample Student and Machine Importance Rankings

- The above table can be interpreted as the following:

  Student s1 thought the most important bear was the black bear, then the polar bear, then brown bear, etc..

  Student s2 thought the most important bear was the polar bear, black bear, brown bear, kodiak bear, etc..

  The machine indicated the most important bear was the black bear, then the polar bear and brown bear, then the sun bear, etc..

- Next, the machine is correlated with all five students to compute its similarity score. Then the students are correlated with each other, excluding any duplicate pairings, to compute their similarity scores. The Spearman's Rank Correlation Coefficient formula is used to calculate the correlations. The result of these correlations is displayed in the table below:

	Machine	Students				
Students	**m**	**s1**	**s2**	**s3**	**s4**	**s5**
s1	0.95					
s2	0.67	0.71				
s3	0.59	0.57	0.82			
s4	0.87	0.71	0.71	0.71		
s5	0.99	0.96	0.64	0.54	0.82	

Table 14.2 - Importance Similarity Scores Example

- The machine *rank-group-consistency score* is computed by averaging the correlation coefficients in column "m" (second column). The class's *rank-group-consistency score* is computed by averaging the correlation coefficients in columns "s1" through "s5". The tolerance value for the class's *rank-group-consistency score* is calculated by taking the variance of columns "s1" through "s5". The result of these three operations is displayed in the table below.

	Machine	Students
Mean	0.81	0.72
Variance		0.02

Table 14.3 - Importance Rank-Group-Consistency Scores Example

- Finally, the *rank-group-consistency score*s are used to evaluate how well the machine could rank a set of entities by importance. In this example, the machine method has exceeded the students score by more than the tolerance, therefor it was able to rank entities by importance better than the average student.

- *Rank-Group-Consistency* score for the class of students indicates how the well the average student did in ranking a set of objects by importance. The table above indicates how well the machine method did against the average student. It can be determined if the machine method did worse than, the same as, or better than the average student in importance ranking, when the *rank-group-consistency* score is used.

14.6.1.4. Rank-Group-Consistency Score Test

The *rank-group-consistency* scores are used to determine to what extent a machine method can rank a set of objects by importance. This is done by comparing the class of students' *rank-group-consistency* score with that of the five machine methods' scores. To make the consistency score comparisons easy

to interpret, they were put into tables. The first table holds the *rank-group-consistency* scores for the survey question, "*Rank the set of objects by importance*". The second table holds the *rank-group-consistency* scores for the survey question, "*Rank the set of objects by how much was learned about them*".

The two *rank-group-consistency* score tables have the same format. Two importance questions were asked for two different reasons. The first reason was that the students might be able to interpret the question, "*Rank the set of objects by how much was learned about them*" better than the question, "*Rank the set of objects by importance*". The second question was rephrased in relation to knowledge, since it is believed that knowledge is largely related with importance. The second reason was to see which question the student rankings would correlate the best with the machine rankings.

A description of the columns and rows of the *rank-group-consistency* scores' table is detailed here. The "Question" row refers to what survey question was asked in an abbreviated form. The "Class Score" row displays the class's *rank-group-consistency* scores for the three surveys that were distributed to the class of college students. The "Tolerance Value" row displays the class's *rank-group-consistency* score variance for each of the three surveys. The "Machine Methods" heading, labels the five machine method's consistency scores. The *Arcs Emanating, Arcs Entering, Arcs In & Out, All Descendants,* and *All Ancestors* rows display each of the five machine method's *rank-group-consistency* scores.

14.6.1.4.1. Machine Method Evaluation with Student Importance Rankings

Importance Rank-Group-Consistency Scores			
Question	*Rank by Importance*		
Article	**Bears**	**Whales**	**Spiders**
Class Score	0.47	0.23	0.12
± **Tolerance Value**	0.10	0.17	0.21
Machine Methods			
Arcs Emanating	0.56	0.53	0.18
Arcs Entering	0.21	0.22	-0.15
Arcs In & Out	0.52	0.49	0.18
All Descendants	0.50	0.42	0.12
All Ancestors	0.65	0.32	0.18

Table 14.4 - Importance Rank-Group-Consistency Score Test

The criterion for this performance evaluation is the machine method's rank consistency with the students compared to the rank consistency among the students themselves. The class consistency score reflects what the average student's *rank-group-consistency* score was. The class's consistency score is the consistency score of each student pairing in the class, excluding any duplicates, averaged together into one score. If the machine method is said to have performed as well as the average student, this means the method was just as consistent as the class of students was in ranking objects by importance.

The *Arcs Emanating* method performed just "as well" or "better than" the average participating student in ranking a set of objects by importance. The *Arcs Emanating* method had a higher *rank-group-consistency* score than the class's consistency score in the whale survey. The method had the same *rank-group-consistency* score as the class's score in the bear and spider surveys. Having a *rank-group-consistency* score "equal to" or "higher than" the class's consistency score, it can be said that the *Arcs Emanating* method can perform the same or better than the average student in ranking objects by importance.

The *Arcs Entering* method performed the "same as" or "worse than" the average student in ranking a set of objects by importance. The *Arcs Emanating* method had lower *rank-group-consistency* scores

than the class's consistency score in the bear and spider surveys. The method had the same *rank-group-consistency* score as the class's score in the whale survey. Having a *rank-group-consistency* score "lower than" the class's consistency score in two of the surveys, it can be said that the *Arcs Emanating* method is at best, fair in ranking a set of objects by importance. The results of this method indicate that arcs entering into an entity have little or nothing to contribute to an entity's importance in this experiment.

The *Arcs In & Out* method performed "as well as" or "better than" the average participating student in ranking a set of objects by importance. The *Arcs In & Out* method had a higher *rank-group-consistency* score than the class's consistency score in the whale survey. The method had the same *rank-group-consistency* score as the class's score in the bear and spider surveys. Having a *rank-group-consistency* score "equal to" or "higher than" the class's consistency score, it can be said that the *Arcs In & Out* method can perform the same or better than the average student in ranking objects by importance.

The *All Descendants* method performed just "as well as" or "better than" the average participating student in ranking a set of objects by importance. The *All Descendants* method had a higher *rank-group-consistency* score than the class's consistency score in the whale survey. The method had the same *rank-group-consistency* score as the class's score in the bear and spider surveys. Having a *rank-group-consistency* score "equal to" or "higher than" the class's consistency score, it can be said that the *All Descendants* method can perform the same or better than the average student in ranking objects by importance.

The *All Ancestors* method performed just "as well as" or "better than" the average participating student in ranking a set of objects by importance. The *All Ancestors* method had a higher *rank-group-consistency* score than the class's consistency score in the bear survey. The method had the same *rank-group-consistency* score as the class's score in the whale and spider surveys. Having a *rank-group-consistency* score "equal to" or "higher than" the class's consistency score, it can be said that the *All Ancestors* method can perform the same or better than the average student in ranking objects by importance.

14.6.1.4.2. Machine Method Evaluation with Student Knowledge Rankings

"Knowledge Gained" Rank-Group-Consistency Scores			
Question	*Rank by Knowledge Gained*		
Article	**Bears**	**Whales**	**Spiders**
Class Score	0.66	0.34	0.32
± **Tolerance Value**	0.05	0.13	0.15
Machine Methods			
Arcs Emanating	0.77	0.53	0.41
Arcs Entering	0.35	0.10	-0.06
Arcs In & Out	0.78	0.47	0.41
All Descendants	0.74	0.52	0.36
All Ancestors	0.68	0.46	0.41

Table 14.5 - "Knowledge Gained" Rank-Group-Consistency Score Test

The criterion for this performance evaluation is the same as it was with the student importance rankings. The machine method's rank consistency with the students is compared to the rank consistency among the students themselves. The class consistency score reflects what the average student's *rank-group-consistency* score was. If the machine method is said to have performed as well as the average student, this means the method was just as consistent as the class of students was in ranking objects by knowledge.

The *Arcs Emanating* method performed just "as well" or "better than" the average student in ranking a set of objects by knowledge. The *Arcs Emanating* method had a higher *rank-group-consistency* score than the class's consistency score in the bear and whale surveys. The method had the same *rank-group-consistency* score as the class's score in the spider survey. Having a *rank-group-consistency* score "equal to" or "higher than" the class's consistency score, it can be said that the *Arcs Emanating* method can perform the same or better than the average student in ranking objects by knowledge.

The *Arcs Entering* method performed "worse than" the average student in ranking a set of objects by "knowledge gained". The *Arcs Entering* method had lower *rank-group-consistency* scores than the class's consistency score in all three surveys. Having a *rank-group-consistency* score "lower than" the class's consistency score, it can be said that the *Arcs Entering* method could not rank a set of objects by "knowledge gained" with accuracy or any similarity to student rankings. The results of this method indicate that arcs entering into entity have little or nothing to contribute to knowledge about an entity in this experiment.

The *Arcs In & Out* method performed just "as well as" or "better than" the average student in ranking a set of objects by "knowledge gained". The *Arcs In & Out* method had a higher *rank-group-consistency* score than the class's consistency score in the bear survey. The method had the same *rank-group-consistency* score as the class's score in the whale and spider surveys. Having a *rank-group-consistency* score "equal to" or "higher than" the class's consistency score, it could be said that the *Arcs In & Out* method can perform the same or better than the average student in ranking objects by "knowledge gained".

The *All Descendants* method performed just "as well as" or "better than" the average student in ranking a set of objects by "knowledge gained". The *All Descendants* method had a higher *rank-group-consistency* score than the class's consistency scores in the bear and whale surveys. The method had the same *rank-group-consistency* score as the class's score in the bear and spider surveys. Having a *rank-group-consistency* score "equal to" or "higher than" the class's consistency score, it can be said that the *All Descendants* method could perform the same or better than the average student in ranking objects by "knowledge gained".

The *All Ancestors* method performed as well as the average student in ranking a set of objects by "knowledge gained". The *All Ancestors* method had the same *rank-group-consistency* score as the class's score in all three surveys. Having a *rank-group-consistency* score "equal to" the class's consistency score in the three surveys, it can be said that the *All Ancestors* method can perform the same as the average student in ranking objects by "knowledge gained".

14.6.2. Determining The Best Importance Ranking Method

The following steps were executed out to determine which machine method could rank a set of objects by importance the best.

- The *rank-individual-consistency* score was defined.
- The *rank-individual-consistency* score formulation was explained.
- An example of how the *rank-individual-consistency* score is computed and used was given.
- The *rank-individual-consistency* score test was performed. This was done to determine which machine method is the best at ranking a set of objects by importance.

14.6.2.1. Rank-Individual-Consistency Score Defined

The *rank-individual-consistency score* determines which machine method can rank a set of objects by importance the best. One way of determining this measure is by comparing the machine's score to the score of each student in the class. The *rank-individual-consistency score* is used to determine if the computer's rankings were similar to a particular student's rankings. This was done by comparing the computer's ranking and a particular student's ranking to the rest of the class. The results of the comparisons are then tabulated as to the number of students that the machine method did "worse than", "the same as" or "better than" in ranking a set of objects by importance. A qualitative measure is then applied to the counts for each machine method to determine which method performed the best in terms of student rank consistency.

A *rank-individual-consistency score* was calculated for the machine and each of the students in the class. The idea was to discover how consistent the machine and the student in question were with the rankings of the rest of the class. One such score was developed for each student in the class other than the one being compared to the machine. These scores were then aggregated by averaging them together. There is one such average for each machine/student comparison. The same thing was done for all students in the class to see how internally consistent the students were in their rankings. The averages indicated how similar the machine was to the students in ranking objects by importance. A high average indicated a high level similarity.

The *rank-individual-consistency score*s of all the machine/student and student/student comparisons were aggregated by counting how many times the machine was "better than", "same as", and "worse than" the students. The machine and student scores were considered the same if they differed by no more than a tolerance value. In this experiment, the tolerance value was the variance of the student's *rank-individual-consistency score*.

The similarity between two sets of rankings can be measured by calculating the Spearman Correlation Coefficient of the two. The Spearman Correlation Coefficient was used since it is specifically constructed to correlate two pairs of rankings. A correlation coefficient with a value close to one indicates a high level of similarity between the two rankings. A correlation value near zero indicates little or no similarity between the two rankings. A correlation value close to negative one indicates rankings that go in opposite directions.

The *rank-individual-consistency* score is used in a different way than the *rank-group-consistency* score. The difference is that the *rank-individual-consistency* score measures the machine method's performance by comparing the method's performance to each student's performance. The *rank-group-consistency* score on the other hand measures how the machine method did compared to the entire class. When the *rank-individual-consistency* score is measured against each student's score, counts are accumulated. These counts indicate if the machine method performed "worse", "the same", or "better"

in ranking a set of objects by importance. The criterion for measuring performance is who was more consistent with the rest of class in ranking a set of objects by importance, the machine method or the student. A qualitative measure is applied to the counts to determine which machine method performed the best in importance ranking.

14.6.2.2. Qualitative Measure Defined

A Qualitative measure was needed to aggregate the machine method's performances over the three surveys into one score. The measure is used to determine which machine method performed the best at ranking a set of objects by importance using the results of the three student surveys. The machine method with the highest qualitative measure would be considered the best method at ranking a set of objects by importance. The criterion for this qualitative measure is the machine method that is the most consistent with the student rankings.

The qualitative measure must take into account when the machine does "worse than", "the same" or "better than" the students. This was done by awarding points to the machine method according to its performance. If the machine method did worse than the student then no points were awarded to its tally. Every time the machine method performed the same as the student then a point was awarded to its tally. Every time the machine method performed better than the student then two points were awarded to its tally. After the points were accumulated per a survey, the tally was divided by the total number of possible points a machine method could earn, to compute a sub score. Finally, when all the sub scores had been computed for all the surveys, they were averaged together to calculate the method's qualitative measure.

The qualitative measure was calculated for each method using the following algorithm.

1. *For each survey perform the following steps.*
   - 1.1 *Multiply the number of times the method performed worse than the students by zero.*
   - 1.2 *Multiply the number of times the method performed the same as the students by one.*
   - 1.3 *Multiply the number of times the method performed better than the students by two.*
   - 1.4 *Add the points calculated in the three previous three steps together.*
   - 1.5 *Divide the total from the previous step by the number of students who participated in the survey multiplied by two. This measure is called the method's survey sub score.*
2. *The method's qualitative score is then computed by averaging its three survey sub scores together.*

14.6.2.3. Rank-Individual-Consistency Score Example

To show how the machine method and student *rank-individual-consistency* scores were calculated and used, a detailed example is given below. The example displays how the *rank-individual-consistency* score is used to determine which machine method performed the best in ranking a set of objects by importance. In the first several steps, only one machine method's *rank-individual-consistency* score is demonstrated to avoid duplicate effort.

- First, five students are given an article about bears to read.
- Second, each student is asked to answer the following question.

  *1) Rank the following objects according to what is the **Most Important** object from the previous article. Rank them in descending order. Give the most important object a "1" and the least important object a rank of "7". All objects must be ranked.*
  ___ polar bear
  ___ brown bear

___ *sun bear*
___ *kodiak bear*
___ *sloth bear*
___ *grizzly bear*
___ *black bear*

- The machine method also reads the same article and answers the above question.
- The five students' and the machine's answers to the above question are numerated and displayed in the table below.

Students/Machine	s1	s2	s3	s4	s5	machine
polar bear	2	1	3	2	2	2
brown bear	3	3	1	3	3	2
sun bear	4	7	7	6	4	4
kodiak bear	5	4	5	7	6	5
sloth bear	6	6	6	4	5	5
grizzly bear	7	5	4	5	7	7
black bear	1	2	2	1	1	1

Table 14.6 - Sample Student and Machine Importance Rankings

- The above table can be interpreted as the following:

  Student s1 thought the most important bear was the black bear, then the polar bear, then brown bear, etc..

  Student s2 thought the most important bear was the polar bear, black bear, brown bear, kodiak bear, etc..

  The machine indicated the most important bear was the black bear, then the polar bear and brown bear, then the sun bear, etc..

- For each of the students (*s1 through s5*) , that were being compared with the machine, the following was done:

  - The machine rankings were correlated, using the Spearman Correlation Coefficient, with each of the students excluding the student to be compared with the machine (*the current student in the loop*).
  - The student that is to be compared with the machine (*the current student in the loop*) has his rankings correlated, using the Spearman Correlation Coefficient, with all the other participating students.

- The numerical result of the above calculation is then recorded in the table below.

	Machine Correlations					Student Correlations				
Method/Student	m1/s1	m1/s2	m1/s3	m1/s4	m1/s5	s1	s2	s3	s4	s5
s1		0.95	0.95	0.95	0.95		0.71	0.57	0.71	0.96
s2	0.67		0.67	0.67	0.67	0.71		0.82	0.71	0.64
s3	0.59	0.59		0.59	0.59	0.57	0.82		0.71	0.54
s4	0.87	0.87	0.87		0.87	0.71	0.71	0.71		0.82
s5	0.99	0.99	0.99	0.99		0.96	0.64	0.54	0.82	

***rank-individual-consistency* score (average)**	**0.78**	**0.85**	**0.87**	**0.80**	**0.77**	**0.74**	**0.72**	**0.66**	**0.74**	**0.74**
tolerance value (variance)						0.03	0.01	0.02	0.00	0.04

Table 14.7 - Sample Machine and Student Correlations for the Rank-individual-consistency score

- The *rank-individual-consistency score*s are then calculated for each of the machine and student runs, columns "m/s1" through "m/s2" for the machine and columns "s1" through "s5" for the students, by taking the average of each column. The *rank-individual-consistency score* in column "m/s1" is the machine score excluding the correlation between the machine and the student "s1". This is the score that will be compared to student "s1's" score.
- The tolerance values were derived from the variance of each student's *rank-individual-consistency score*.
- The five machine *rank-individual-consistency score*s are then compared to their corresponding student scores (i.e., *machine score m/s1 is compared to student score s1*). These comparison pairings are displayed in the table below. Notice that the machine score from the above table in column "m/s1" is positioned in column "s1" in the below table. The scores were considered the same if they differed by no more than their corresponding tolerance value.

Rank-Individual-Consistency Scores	**s1**	**s2**	**s3**	**s4**	**s5**
student	0.74	0.72	0.66	0.74	0.74
tolerance value	0.03	0.01	0.02	0.00	0.04
machine method	0.78	0.85	0.87	0.80	0.77

Table 14.8 - Sample Machine and Student Rank-individual-consistency score Comparisons

- The machine did better than students "s1", "s2", "s3" and "s4" in ranking a set of objects by importance when the criterion was rank similarity with the rest of the students excluding the student the machine was compared with. The machine did as well as student "s5" in ranking objects by importance when the criterion was rank similarity with the rest of the students.
- To demonstrate how one machine method is determined to be better than another, a second machine method's *rank-individual-consistency* scores were added to the above table to create the table below. The machine method whose consistency score was just calculated will be referred to as *method one*. The new machine method being introduced will referred to as *method two*.

Rank-Individual-Consistency Scores	**s1**	**s2**	**s3**	**s4**	**s5**
student	0.74	0.72	0.66	0.74	0.74
tolerance value	0.03	0.01	0.02	0.00	0.04
machine method one	0.78	0.85	0.87	0.80	0.77
machine method two	0.81	0.71	0.77	0.73	0.72

Table 14.9 - Sample Machine and Student Rank-Individual-Consistency Score Comparisons

- Using the above table, the number of times a machine method performed "worse than", "the same as", or "better than" is accumulated into the table below.

Consistency Score Student Comparison Counts			
Question	*Rank the objects by importance*		
Article	**Bears**		
Number of times method did worse, the same, or better than the students.	**worse**	**same**	**better**
Machine Method			
Machine Method One	0	0	5
Machine Method Two	1	2	2

Table 14.10 - Sample Consistency Score Student Comparison Counts

- The table above indicates that the machine *method one* did better than all five students in this example. In other words, it had a higher score than all five students. *Method two* on the other hand, performed worse than one student, the same as two students, and better than two students in importance ranking.
- The qualitative measure is applied to determine which machine method performed the best. The qualitative measure will use the counts that were accumulated in the above table to gauge a method's performance. The calculations for the qualitative measure are displayed in the table below.

Importance Ranking Qualitative Measure Calculations					
Question	*Rank the objects by importance*				
Article	**Bears**				
Number of times method did worse, the same, or better than the students.	**worse**	**same**	**better**	**tally**	**Qualitative Measure**
Machine Method					
Machine Method One	0 x 0	0 x 1	5 x 2	10	10 / 10 = 1.0
Machine Method Two	1 x 0	2 x 1	2 x 2	6	06 / 10 = 0.6

Table 14.11 - Sample Importance Ranking Qualitative Measure Calculations

- Observing the qualitative measures pictured in the table above. It can be seen that machine *method one* performed better than machine *method two*. This can be attested, since method one's score of 1.0 is higher than method two's score of 0.6. This indicates that method one is better than method two in ranking a set of objects by importance. The criterion for the performance evaluation is similarity to the student importance rankings.

14.6.2.4. Rank-Individual-Consistency Score Test

The *rank-individual-consistency* scores are used in conjunction with a qualitative measure to determine which machine method can rank a set of objects by importance the best. This was done by comparing each machine method's *rank-individual-consistency* score with its corresponding student's consistency score. Counts were accumulated for every time a machine method had a *rank-individual-consistency* score "lower than", "equal to", or "higher than" a student's consistency score. The counts were accrued into a table, under the headings "worse", "same", and "better" respectively. The different counts can be used with the help of a qualitative measure, to determine which machine method performed the best at ranking a set of objects by importance. The machine method with the highest qualitative score is considered the best.

Two tables were created, to handle the results of the two questions dealing with importance. The first table holds the *rank-individual-consistency* score counts for the survey question, "*Rank the set of objects by importance*". The second table holds the *rank-individual-consistency* score counts for the survey question, "*Rank the set of objects by how much was learned about them*". The two tables have the same format.

A description of the columns and rows of the *rank-individual-consistency* score count table is detailed here. The "Question" row refers to what survey question was asked. The "Article" row refers to what survey was used in the evaluation. The "Machine Method" heading indicates the start of machine method results. The column labeled with "worse", indicates the number of students a machine method had a lower *rank-individual-consistency* score than. The column labeled with "same", counts the number of students whose consistency score was the same as the machine method's score. The column labeled with "better", indicates the number of students a machine method had a higher consistency score than. The column labeled with "Qualitative Score" displays each machine method's qualitative score. The rows *Arcs Emanating, Arcs Entering, Arcs In & Out, All Descendants*, and *All Ancestors* displays the accumulated counts for each machine method.

14.6.2.4.1. Machine Method Evaluation with Student Importance Rankings

***Rank-Individual-Consistency* Score Count Comparisons**										
Question	*Rank the objects by importance*									
Article	**Bears**			**Whales & Dolphins**			**Spiders**			
Number of times method did worse, the same, or better than the students.	**worse**	**same**	**better**	**worse**	**same**	**better**	**worse**	**same**	**better**	**Qualita -tive Score**
Machine Method										
Arcs Emanating	2	4	4	0	4	7	1	16	3	**0.66**
Arcs Entering	9	1	0	2	3	6	12	8	0	**0.31**
Arcs In & Out	3	5	2	0	5	6	1	16	3	**0.59**
All Descendants	3	5	2	0	5	6	3	14	3	**0.57**
All Ancestors	0	2	8	1	6	4	1	16	3	**0.70**

Table 14.12 - Rank-Individual-Consistency Count for Importance Rankings

The *All Ancestors* method was the best machine method in ranking objects by importance. This was concluded, since the *All Ancestors* method's qualitative score was the highest when compared to other method's scores. This indicates that the *All Ancestors* method had the least number of students who performed better than it and the most number of students that performed the same or worse than it. Performance was based upon consistency with the rest of the class in ranking a set of objects by importance.

The *Arcs Entering* method was the worst machine method in ranking objects by importance. This was concluded, since the *Arcs Entering* method's qualitative score was the lowest when compared to other method's scores. Meaning that the *Arcs Entering* method had the most number of students who performed better than it and the least number of students that performed the same or worse than it. Performance was based upon consistency with the rest of the class in ranking a set of objects by importance.

14.6.2.4.2. Machine Method Evaluation with Student "Knowledge Gained" Rankings

***Rank-Individual-Consistency* Score Count Comparisons**	
Question	*1) Rank the objects by how much was learned about them.*

Article	Bears			Whales			Spiders			
Number of times method did worse, same, or better than the students.	worse	same	better	worse	same	better	worse	same	better	Qualita -tive Score
Machine Method										
Arcs Emanating	0	3	7	0	5	6	0	15	5	**0.75**
Arcs Entering	9	1	0	8	3	0	18	2	0	**0.08**
Arcs In & Out	0	2	8	0	5	6	0	15	5	**0.77**
All Descendants	0	5	5	0	5	6	2	13	5	**0.70**
All Ancestors	4	4	2	1	5	5	0	15	5	**0.57**

Table 14.13 - Rank-Individual-Consistency Count for "Knowledge Gained" Rankings

The *Arcs In & Out* method was the best machine method in ranking objects by "knowledge gained." This was concluded, since the *Arcs In & Out* method's qualitative score was the highest when compared to other method's scores. Meaning that the *Arcs In & Out* method had the least number of students who performed better than it and the most number of students that performed the same or worse than it. Performance was based upon consistency with the rest of the class in ranking a set of objects by "knowledge gained".

The *Arcs Entering* method was the worst machine method in ranking objects by "knowledge gained." This was concluded, since the *Arcs Entering* method's qualitative score was the lowest when compared to other method's scores. Meaning that the *Arcs Entering* method had the most number of students who performed better than it and the least number of students that performed the same or worse than it. Performance was based upon consistency with the rest of the class in ranking a set of objects by "knowledge gained".

14.6.3. Investigators View of the Machine's Importance Rankings

It appeared that a machine provided with an importance ranking method could rank objects by importance very well. Looking at the machine's importance rankings, it seemed the machine rankings made sense after it had read the natural language text. For example whales that were mentioned and described the most were located at the top of the rankings. Another good observation about the machine's importance rankings is that it never gave the same rank to more than one fourth of the objects being ranked. This meant the machine method was able too different between objects at a reasonable level.

One thing that could be improved in the machine importance ranking is the differential in rankings between animal subsets and animal sub species. At times, the importance method ranked species subsets along with the sub species. For example, the machine method ranked "mother bear" higher than and along with "kodiak bear" and "grizzly bear". There should have been some ranking difference between the specie subsets and the sub species.

In future work, this problem of mixing specie subsets and sub-species in importance ranking could be solved in a couple of ways. One way would be to create a different taxonomy structure that would exclude the specie subsets from the importance rankings. This would also fix the categorization problems that were experienced earlier in this experiment. Another solution would be to provide more information in the form of natural language text or in the defined dictionary.

14.7. Conclusion

To show that a machine has the conceptualization ability of ranking related entities by importance was a complex process to execute. First, different machine methods had to be developed to rank entities by importance. Second, tests had to be found that could evaluate the results of the machine methods. Next, students had to be surveyed, so there was data to gauge the accuracy of the machine methods in ranking related entities by importance. Importance ranking accuracy had to be defined and quantified. Finally, the results of the evaluations were collected and summarized to determine if a machine with the appropriate method could rank related entities by importance with any degree of accuracy.

The five different machine methods that were developed to rank entities by importance were:

- *Emanating Arcs Method* - Rank the entities by the number of arcs that emanate from them.
- *Entering Arcs Method* - Rank the entities by the number of arcs that enter into them.
- *Arcs In & Out Method* - Rank the entities by the number of arcs that emanate from them plus the number of arcs that enter into them.
- *All Descendants Method* - Rank the entities by the number of arcs that emanate from them plus the number of arcs that emanate from their primary descendants.
- *All Ancestors Method* - Rank the entities by the number of arcs that emanate from them plus the number of arcs that emanate from their primary ancestors.

The *rank-group-consistency score* was developed to evaluate to what extent a machine method could rank a set of objects by importance. The score essentially computes the similarity of machine and human rankings. The score was computed for both the machine and the class of students. The machine *rank-group-consistency score* entailed correlating the machine importance rankings with each student's rankings and averaging the coefficients into one score. The class *rank-group-consistency score* involved correlating each student's importance rankings with every other student's rankings, excluding duplicated pairings, and averaging the coefficients into one score. A tolerance value was also calculated by taking the variance of the class's correlation coefficients. The extent that a machine can rank a set of objects was measured by comparing the machine's *rank-group-consistency score* with the class's *rank-group-consistency score*. The scores were considered the same if they differed by no more than the tolerance value.

The *rank-individual-consistency* score was developed to determine which machine method could rank a set of objects by importance the best. The *rank-individual-consistency* score is used in a different way than the *rank-group-consistency* score. The difference is that the *rank-individual-consistency* score measures the machine method's performance by comparing the method's performance to each student's performance. The *rank-group-consistency* score on the other hand, measures how well the method did compared to the entire class. When the *rank-individual-consistency* score is measured against each student's score, counts are accumulated. These counts indicate if the machine method performed "worse than", "the same as", or "better than" a group of students in ranking a set of objects by importance. The criterion for performance was, whom was more consistent with the rest of class in ranking a set of objects by importance, the machine method or the student. The counts were then used in calculating a qualitative measure. This qualitative measure determined which machine method performed the best in ranking a set of objects by importance.

Three surveys were taken from a college student population to evaluate the performance of the machine importance methods. In each survey the students were handed a packet that consisted of relevant definitions, an article, and a set of questions. The relevant definitions supplied each student with standard definition of importance and an example of the concept. The article received by the students was about specific entities that they were to read about and to use as a reference in answering questions. The two questions the students were given, "*Rank the following objects according to what you learned from the article*" and "*Rank the following object according to what is the Most Important*

object from the previous article," were used to capture the student importance response. The data derived from the student responses were then used to validate the results of the machine methods.

The *rank-individual-consistency* score in conjunction with the qualitative measure determined which machine method performed the best at ranking a set of objects by importance. In this experiment the *rank-individual-consistency* score indicated that the *All Ancestors* method was the best, at ranking a set of objects by importance. The criterion for this performance evaluation is student rank consistency. This was due to the fact that this machine method performed worse than least number of students and better than the greatest number of students.

Determining to what extent a machine can rank a set of objects by importance was a twofold process. First, the best performing machine method had to be selected. This was done using the *rank-individual-consistency* score and it picked the *All Ancestors* method. Second, the *All Ancestors* method's ranking performance was evaluated in terms of the average student. In this experiment, the *rank-group-consistency* score was used as a gauge to evaluate machine performance. The *All Ancestors* method can rank a set of objects by importance as well as or better than the average student. The average student being defined as a student who participated in the survey and was compared with the machine method and his fellow students in ranking objects by importance.

A secondary experiment was tried along with importance ranking experiment. It was thought if the importance rank question was rephrased in terms of knowledge, the students might better understand the question. Knowledge was used as a ranking mechanism, since it is a large contributor to entity importance. With this in mind, the students were asked a second importance ranking question, "*Rank the objects by the knowledge learned (gained) from the article*".

The *rank-individual-consistency* scores in conjunction with the qualitative measure were used again to determine which machine method performed the best at ranking a set of objects by "knowledge gained". In this experiment the *rank-individual-consistency* score indicated that the *Arcs In & Out* method was the best at ranking a set of objects by "knowledge gained". The criterion for this performance evaluation was student rank consistency. This was due to the fact that this machine method performed worse than least number of students and better than the greatest number of students.

The *rank-group-consistency* scores of the *Arcs In & Out* method were used to determine to what extent a machine could rank a set of objects by "knowledge gained". In this experiment, the *rank-group-consistency* scores indicated that a machine could rank a set of objects by "knowledge gained" as well as or better than the average student. The criterion for this performance evaluation was student rank consistency.

The experiment results of both questions looked at simultaneously, lead to a couple of observations. If the class's *rank-group-consistency* scores are looked at, it appears that the students were more consistent with each other when answering the "knowledge gained" question than when answering the "importance" question. This is attested by their higher consistency scores and lower variances when they were asked to rank a set of objects by "knowledge gained" from the article. This indicated that the students agreed together more as whole what knowledge was, compared to what they thought importance was. It also followed that the machine method's consistency scores were also higher. It seems that the students did understand the importance ranking question better, when asked in terms of knowledge.

15. Best Example Entities

15.1. Introduction

A *Best Example* entity is an entity that gives the best general description of its category and there is usually one in every category [Lackoff, 1987]. It also can be the entity that is used as a stereotypical example, a central member of its category and its features are the typical features of its category [Smith, Medin, 1981; Sowa, 1984]. The ability to develop a prototype that represents the *best example* of a category is an essential tool in conceptualization. *Best example* entities are essential to conceptualization for the following reasons.

- They can give us a quick definition of a category and the attributes contained within that category.
- Lackoff elaborates that *best examples* are used as prototypes for inferencing:

  > "... In many cases, prototypes act as cognitive reference points of various sorts and form the basis for inferences. The study of human inference is part of the study of human reasoning and conceptual structure; hence, those prototypes used in making inferences must be part of a conceptual structure." [Lackoff, 1987]

- Rosch and Mervis [1975] explain that *best examples* allow for efficient cognition:

  > "... The present study has shown that empirically defined prototypes of natural categories are just those items with highest cue validity. Such a structure of categories would, in fact, appear to provide the means for maximally efficient processing of categories."

- It is a category reference point that must be seen as in relation to other entities within the category. [Rosch, 1975]
- They are used as a measure of class membership for other entities within its class or whether to classify a non-member entity within its class. [Smith, Medin, 1981; Sowa, 1984]

It is critical that *best example* determination be included in the evaluation of machine conceptualization, since it provides a cognitive reference point, category structure, measure for class membership, standard for comparison, and a base for inferencing.

15.2. Purpose

The goal of this chapter is to determine to what extent a machine method can rank set of objects by *best example*. This is done by demonstrating the degree of similarity between machine and human *best example* rankings. This process is elaborated upon by:

- Illustrating the process involved in choosing an entity that represents the *best example* of a selected category
- Explaining the machine method's *best example* output
- Discussing the student survey's *best example* question
- Explicating the measures used to evaluate a machine method's performance
- Validating machine output using the student survey results

With the previous steps executed, the quality and human similarity of machine method's *best example* selection can be evaluated to some degree.

A secondary goal of this chapter is to determine which machine method out of the four tried, performed the best in ranking entities by *best example*.

15.3. Process

The process of ranking related entities by *best example* criteria was performed in four steps. The first step gathered a subset of entities to be ranked. The second step, computed a base *best example* score for all the entities collected in the previous step. The next step tried four different machine methods to calculate *best example* scores. The fourth step used an algorithm to rank the subset of selected entities from the previous step in descending order according to their *best example* scores. The subset selection process, base score computation, four different scoring methods, and ranking algorithm are reviewed in the following sections.

15.3.1. Subset Selection Process

The subset selection process used in *best example* ranking is the same process that was used in the "*Rank Entities by Importance*" process. For brevity it is summarized below.

The subset selection process creates an array of entities to be ranked by performing the following tasks. First the process inquires the user for what subset of entities he or she would like to be ranked. Next, the user then inputs an entity that is contained within the semantic net. The process then finds the user specified entity in the net and gathers all of its primary descendants. Two algorithms are involved in collecting a subset of entities and are summarized here. The first algorithm, *Find Entity*, finds the user selected entity in the net. The second algorithm, referred to as *Collect Entities*, gathers the primary descendants of the entity returned by the *Find Entity* algorithm and places them into an array.

For example, using figure 15.1 displayed below, the subset of entities contained in the whale subset would be: toothed whale, orca whale, blue whale, and sperm whale. The user would have had typed in "whale" and the *Find Entity* algorithm would have followed the paths "*entity INCLUDES whale*" and "*mammal INCLUDES whale*", and returned a pointer to "whale". The *Collect Entities* algorithm would than have collected toothed whale, orca whale, blue whale, and sperm whale entities when it traversed the following paths: "*whale HAS SUBSET toothed whale*", "*whale INCLUDES orca whale*", "*whale INCLUDES blue whale*", and "*toothed whale INCLUDES sperm whale*".

Figure 15.1 - Sample Semantic Net Used for Subset Collection

15.3.2. Base Score Calculation

All four machine methods that were evaluated in ranking entities by *best example* criteria used the same "base" *best example* calculation process. The base calculation rules and conditions that were applied to each entity to be ranked are listed below:

1) Multiply the number of "ISA" arcs leading to the *best example* category entity by two and add it to the *best example* score.
2) Add the number of "SUBSET OF" arcs leading to the *best example* category entity to the *best example* score.

3) Add the number of "INSTANCE OF" arcs leading to the *best example* category entity to the *best example* score.

4) Add the number of similar relationships between the selected *best example* category entity and the descendent entity, excluding "INSTANCE", "INCLUDES" and "HAS SUBSET" relations, to the *best example* score.

5) Add the number of similar relationships between the selected *best example* category entity "HAS SUBSET" children and the descendent entity to the *best example* score. For example, *"orca whale have sharp teeth"* relationship matches with the whale's subset *"toothed whale"* relationship *"toothed whale have teeth"*, where whale is the orca whale's parent.

6) Subtract the number of opposing relationships between the selected *best example* category entity and the descendent entity, excluding "INSTANCE", "INCLUDES" and "HAS SUBSET" relations, from the *best example* score.

7) Subtract the number of opposing relationships between the selected *best example* category entity "HAS SUBSET" children and the descendent entity from the *best example* score.

The above base *best example* rules try to measure the strength between the candidate entity and its category root. They also try to calculate how much the candidate entity has in common with its sibling entities to determine its stereotypically among its peers.

Figure 15.2 - Semantic Net Used For Base Best Example Calculation

An example of a base *best example* calculation is given below using the sample semantic net pictured above for input. The example results are displayed in the following table. The orca whale will be used to demonstrate how the base score is calculated.

Base Score Example Calculation

Rule	Relationship	Quantity	Multiplier	Score
1	orca whale ISA whale	1	2	2
4	orca whale HAVE black fins	1	1	1
5	orca whale HAVE sharp teeth	1	1	1
Base Score Total				***4***

Example 15.1 - Base Score Example Calculation

15.3.3. Four Scoring Methods

The process of ranking *best example* entities, encompassed counting similar and dissimilar relationships between entities. This approach was used because:

- The prototype is the central member of its category and its features are the typical features of its category. [Smith, Medin, 1981; Sowa, 1984; Mazlack]
- *Best examples* have attributes that overlap those of other members of the category. They also have attributes that don't overlap with members of other categories. [Rosch, Mervis, 1975]
- Rosch and Mervis explicated that *best examples* are members of a category that contain the common features and attributes within that category. [Rosch, Mervis, 1975]

These experts suggest that a *best example* entity is the one that has typical features of its corresponding category and not the typical features of its pier categories. For example, the *best example* of a cat would have the common features that all cats have and not contain the common features that dogs, birds, horses, etc., would have.

The four scoring methods evaluated to determine which one simulated human *best example* rankings the best, were:

- *Minimal Method*
- *Combination Method*
- *Multiplier Method*
- *Multiplier & Combination Method*

The four different methods were used to decide which entity characteristics and attributes contributed the most in predicting human *best example* rankings. The entity attributes considered in this study were the number of arcs emanating from an entity, the quantity of similar and dissimilar relationships between the entity and other entities, and whether those relationships were within or without the entity's respective category. Each method's description, algorithm and sample calculation (*using the semantic net pictured below*) are detailed below.

Figure 15.3 - Semantic Net Used For Best Example Calculations

15.3.3.1. Minimal Method

The *Minimal Method* is the simplest of the *best example* algorithms. This method uses only the *Base Score Calculation* to compute each entity's *best example* score. The rules for this method have already been explained in the *Base Score Calculation* section above, but the algorithm is stated below. The algorithm consists of two procedures, *Determine Strength* and *Compare Relations*. The *Determine Strength* procedure determines the strength of the relation between the entity and its category node, and appropriates the proper number of points to the entity's *best example* score. The *Compare Relations* procedure awards' points when the entity has the same relationships as the other entities contained in its category and subtracts points when the actions of the relationships are opposing.

For each of the entities to be ranked do:

- Accumulate the number of "ISA" arcs leading to the superordinate node and multiply the total by 2.
- Accumulate the number of "SUBSET OF" arcs leading to the superordinate node.
- Accumulate the number of "INSTANCE OF" arcs leading to the superordinate node.
- Accumulate the number of similar relationships between the superordinate node and the entity.
- Accumulate the number of similar relationships between the superordinate node's subsets and the entity.
- Decrement the number of opposing relationships between the superordinate node and the entity.
- Decrement the number of opposing relationships between the superordinate node's subsets and the entity.

Algorithm 15.1 - Minimal Method (summation)

```
exp_score  /*Best Example Score*/
entity     /*Current entity*/
```

```
    relation[] /*Arcs emanating from the current entity*/
    children[] /*Current entity's children connected by relation*/
    category   /*Categorical Entity Node that "entity" is a member of*/

    /*Determine Strength Procedure*/
    entity.exp_score = 0
    i = 0
    While i < entity.relation_quantity Do
        If entity.children[i] = category Then
            If entity.relation[i] = "ISA" Then
                entity.exp_score = entity.exp_score +
                            relation[i].quantity * 2
            End If
            If entity.relation[i] = "SUBSET OF" Then
                entity.exp_score = entity.exp_score +
                            relation[i].quantity
            End If
            If entity.relation[i] = "INSTANCE OF" Then
                entity.exp_score = entity.exp_score +
                            relation[i].quantity
            End If
        End If
        i = i + 1
    End While

    /*Call the CompareRelations Procedure*/
    entity.CompareRelations(category)
```

Algorithm 15.2 - Determine Strength Procedure

The *Determine Strength* procedure is called by the entity whose *best example* score is to be calculated. It is passed the category node the entity is a member of. First, the procedure goes through the entity's relationships searching for the category node. When the category node is found, the type of relation between the entity and the category is identified and the appropriate number of points is awarded to the *best example* score according to the relation type. If the relation is a *class membership* ("ISA") type then the relation quantity from the entity to the category node is multiplied by two and added to the entity's *best example* score. Otherwise, If the relation is a *subset* ("SUBSET OF") or *instance* ("INSTANCE OF") type then the number of relations from the entity to the category node is added to the entity's *best example* score. Finally, the *Determine Strength* procedure calls the *Compare Relations* procedure to finish the base *best example* calculation. The *Compare Relations* procedure is displayed and explained below.

```
    exp_score  /*Best Example Score*/
    entity     /*Current entity*/
    relation[] /*Arcs emanating from the current entity*/
    children[] /*Current entity's children connected to  relation*/
    neg_sw     /*Indicates a positive or negative relation*/
    category   /*Categorical Entity Node the "entity" is member of*/

    /*Compare Relations Procedure*/
    CompareRelations(category)
    Begin
        i = 0
        While i < category.relation_quantity Do
            /*Compare relations amongst categories subsets*/
            If category.relation[i] = "HAS SUBSET" Then
                entity.CompareRelations(category.children[i])
```

```
            End If
            j = 0
            While j < entity.relation_quantity Do
                If  (entity.relation[j] = "SUBSET OF") And
                    (entity = INSTANCE) Then
                    continue
                End If
                /*Find common relationships*/
                If  (entity.relation[j] = category.relation[i]) And
                    (entity.children[j] = category.children[j]) Then
                    /*Determine polarity of relations*/
                    If entity.relation[j].neg_sw =
                category.relation[i].neg_sw Then
                        entity.exp_score = entity.exp_score + 1
                    Else
                        entity.exp_score = entity.exp_score - 1
                    End If
                End If
                j = j + 1
            End While
            i = i + 1
        End While
    End
```

Algorithm 15.3 - Compare Relations Procedure

The *Compare Relations* procedure contributes to an entity's *best example* score by comparing the entity's relationships with that of its category node and the category node's subsets. This is done in two steps. The first step recursively calls the *Compare Relations* procedure passing along each of the category's subset entities. The second step then compares the passed in category's relationships with that of the calling entity's relationships. If the two relationships match and are of the same polarity then a point is added to the entity's *best example* score. If the two relationships are of opposing polarity then a point is subtracted from the score. Subset relationships where the calling entity is an instance of the category node are excluded from the scoring logic.

An example of how the *Minimal Method* works in calculating the orca whale's *best example* of a whale score is illustrated below. The semantic net displayed in figure 15.3 is used for input. To the left of each awarded point is the matching rule from the *15.3.2 Base Score Calculation* section along with the corresponding relationship.

***Minimal Method* Example**

Rule	Relationship / Rule Match	Quantity	Multiplier	Score
1	orca whale ISA whale *ISA relation is of membership type*	1	2	2
4	orca whale HAVE black fins *matches with* whale HAVE fins	1	1	1
5	orca whale HAVE sharp teeth *matches with* toothed whale HAVE teeth	1	1	1
Best Example Score				***4***

Example 15.2 - Minimal Method Calculation

15.3.3.2. Combination Method

The *Combination Method* is the same as the *Minimal Method* except that it adds the number of arcs that emanate from an entity to its *best example* score. There is no algorithm to count the number of arcs that emanate from an entity, since the counting of arcs is done during the formation of the semantic net. The value of the *best example* score using the *Combination Method* is the result of the *Base Calculation Score* plus the number of arcs that emanate from the entity.

For each of the entities to be ranked do:

- Accumulate the number of "ISA" arcs leading to the superordinate node and multiply the total by 2.
- Accumulate the number of "SUBSET OF" arcs leading to the superordinate node.
- Accumulate the number of "INSTANCE OF" arcs leading to the superordinate node.
- Accumulate the number of similar relationships between the superordinate node and the entity.
- Accumulate the number of similar relationships between the superordinate node's subsets and the entity.
- Decrement the number of opposing relationships between the superordinate node and the entity.
- Decrement the number of opposing relationships between the superordinate node's subsets and the entity.
- Accumulate the number of arcs that emanate from the entity.

Algorithm 15.4 - Combination Method (summation)

An example of how the *Combination Method* works in calculating the *orca whale's best example* of a whale score is illustrated below. The semantic net displayed in figure 15.3 is used for input. To the left of each awarded point is the matching rule from the *15.3.2 Base Score Calculation* section along with the corresponding relationship.

***Combination Method* Example**				
Rule	**Relationship / Rule Match**	**Quantity**	**Multiplier**	**Score**
1	orca whale ISA whale *ISA relation is of membership type*	1	2	2
4	orca whale HAVE black fins *matches with* whale HAVE fins	1	1	1
5	orca whale HAVE sharp teeth *matches with* toothed whale HAVE teeth	1	1	1
Base Score (Minimal Method)				***4***
Entity's Emanating Arcs	orca whale ISA whale	1	1	1
	orca whale HAVE black fins	1	1	1
	orca whale HAVE sharp teeth	1	1	1
	orca whale IS smart	1	1	1
	orca whale kill dolphin	1	1	1

Total Number of Emanating Arcs	***5***
Best Example Score (base score + emanating arcs)	***9***

Example 15.3 - Combination Method Calculation

15.3.3.3. Multiplier Method

The *Multiplier Method* is a calculation that involves several algorithms to compute an entity's *best example* score. The following steps are performed by this method to calculate the *best example* score for each considered entity:

- The number of primary children the category's primary parent has, is counted using the *Count Primary Children* algorithm.
- The entity's base score is calculated using the *Base Score Calculation* and is used to initialize its *best example* score.
- The entity's *best example* score is then multiplied by the number of its category's primary parent's children. This is done to normalize the entity's *best example* score for later processing, since relationship numbers of other categories will be used in the computation of the entity's *best example* calculation.
- The number of relationships the entity has in common with other entities in its category is added to the entity's *best example* score. This is done using the *Count Common Relationships* algorithm.
- The number of relationships the entity has in common with other entities in other the categories besides its own category is subtracted from the entity's *best example* score. This is done using the *Count Common Relationships* algorithm.
- The number of relationships the entity does not have in common with other entities in other the categories besides its own is added to the entity's *best example* score. This is done using the *Count UnCommon Relations* algorithm.

The *Count Primary Children*, *Count Common Relationships*, and *Count UnCommon Relationships* algorithms used by the *Multiplier Method* are discussed in detail below.

15.3.3.3.1. Count Primary Children Algorithm

The *Count Primary Children* algorithm returns the number of primary children the calling entity has. The algorithm accumulates the number of "INCLUDES" and "HAS SUBSET" relations into a total that it returns. The algorithm is detailed below.

```
entity     /*Calling entity*/
relation[] /*Arcs emanating from the calling entity*/

/*Count Primary Children Algorithm*/
CountPrimaryChildren()
Begin
    total = 0
    i = 0
    While i < entity.relation_quantity Do
        If  (entity.relation[i] = "INCLUDES") Or
            (entity.relation[i] = "HAS SUBSET") Then
            total = total + 1
        End If
        i = i + 1
    End While
```

```
    Return (total)
End
```

Algorithm 15.5 - Count Primary Children

15.3.3.3.2. Count Common Relationships Algorithm

The *Count Common Relationships* algorithm counts the number of relationships that two entities have in common. This is done by looping through the first entity's relationships and comparing each one to the second entity's relationships. Every time there is a match between the two entities an accumulator is incremented by one. When the first entity is done looping through its relationships the accumulator is returned to the calling procedure. Category membership relation types such as class membership, subsets, and instances are exempt from the counting process. This exemption is enacted, since all the entities in the semantic net will have at least one form of membership if not more. One other exception to incrementing the accumulator, is that duplicate relationships are not counted twice. The algorithm is displayed below.

```
entity1    /*The entity whose common relationships are counted*/
entity2    /*The entity whose relationships are being compared*/
relation[] /*Arcs emanating from the calling entity*/
children[] /*Entity's children entities connected by its relations*/

/*Count Common Relationships Algorithm*/
CountCommonRelationships()
Begin
    total = 0
    i = 0
    /*Exempt Membership Relations From Accumulator*/
    While i < entity1.relation_quantity Do
        If  (entity1.relation[i] = "INCLUDES") Or
            (entity1.relation[i] = "HAS SUBSET") Or
            (entity1.relation[i] = "INSTANCE") Or
            (entity1.relation[i] = "ISA") Or
            (entity1.relation[i] = "INSTANCE OF") Or
            (entity1.relation[i] = "SUBSET OF") Then
            Continue
        End If
        j = 0
        While j < entity2.relation_quantity Do
            /*Identify Common Relationships*/
            If  (entity1.relation[i] = entity2.relation[j]) And
                (entity1.children[i] = entity2.children[j]) Then
                total = total + 1
                break/*Prevent Counting Of Duplicate Relationships*/
            End If
            j = j + 1
        End While
        i = i + 1
    End While
    Return (total)
End
```

Algorithm 15.6 - Count Common Relationships

15.3.3.3.3. Count UnCommon Relationships Algorithm

The *Count UnCommon Relationships* algorithm is similar to that of the *Count Common Relationships* algorithm, except that it counts the number of uncommon relationships between two entities. This is

done by looping through the first entity's relationships and comparing each one to the second entity's relationships. Every time the first entity has a relationship the second entity does not have an accumulator is incremented by one. When the first entity is done looping through all its relationships the accumulator is returned to the calling procedure. Category membership relation types such as class membership, subsets, and instances are exempt from the counting process. This exemption is enacted, since all the entities in the semantic net will have at least one form of the membership if not more. The algorithm is iterated below.

```
entity1    /*The entity whose uncommon relationships are counted*/
entity2    /*The entity whose relationships are being compared*/
relation[] /*Arcs emanating from the calling entity*/
children[] /*Entity's children entities connected by its relations*/

/*Count UnCommon Relationships Algorithm*/
CountUnCommonRelationships()
Begin
    total = 0
    i = 0
    /*Exempt Membership Relations From Accumulator*/
    While i < entity1.relation_quantity Do
        If  (entity1.relation[i] = "INCLUDES") Or
            (entity1.relation[i] = "HAS SUBSET") Or
            (entity1.relation[i] = "INSTANCE") Or
            (entity1.relation[i] = "ISA") Or
            (entity1.relation[i] = "INSTANCE OF") Or
            (entity1.relation[i] = "SUBSET OF") Then
            Continue
        End If
        j = 0
        flag = False
        While j < entity2.relation_quantity Do
            /*Identify Common Relationships*/
            If  (entity1.relation[i] = entity2.relation[j]) And
                (entity1.children[i] = entity2.children[j]) Then
                flag = True
                break
            End If
            j = j + 1
        End While
        /*The Second Entity Did Not Have A Matching Relationship*/
        If flag = False Then
            total = total + 1
        End If
        i = i + 1
    End While
    Return (total)
End
```

Algorithm 15.7 - Count UnCommon Relationships

15.3.3.3.4. Multiplier Method Summary

The *Multiplier Method* using the *Base Score Calculation, Count Primary Children, Count Common Relations,* and *Count UnCommon Relationships* computes an entity's *best example* score. This method emphasizes the features an entity has in common with other entities in its category while de-emphasizing those features the entity has in common with other entities outside its category. This

method tries to determine if an entity's common and uncommon relationships within and without its category can emulate a human's *best example* behavior.

For each of the entities to be ranked do:

- Calculate the entity's base score *(minimal method)*.
- Count the entity's primary children.
- Set the entity's best example score equal to the entity's base score multiplied by the number of its primary children.
- Increment the entity's best example score by the number of relationships it has in common with other entities in its category.
- Decrement the entity's best example score by the number of relationships it has in common with other entities that are not in its category but are at a level below it.
- Increment the entity's best example score by the number of relationships it does **not** have in common with other entities that are not in its category but are at a level below it.

Algorithm 15.8 - Multiplier Method (summation)

An example of how the *Multiplier Method* works in calculating the *orca whale's best example* of a whale score is illustrated below. The semantic net displayed in figure 15.3 is used for input. To the left of each awarded point is the matching step used from the algorithm along with the corresponding relationship.

Multiplier Method **Example**				
Rule	**Relationship / Rule Match**	**Quantity**	**Multiplier**	**Score**
Base Score *(from example 15.2)*				***4***
Category's Parent's Primary Children	mammal INCLUDES whale	1	1	1
	mammal INCLUDES dolphin	1	1	1
Number of the Category's Parent's (mammal) Primary Children				***2***
	Base Score × Category's Parent's Primary Children	4	2	***8***
Common Within Category	orca whale HAVE sharp teeth *matches with* toothed whale HAVE teeth	1	1	1
Common Without Category	orca whale IS smart *matches with* dolphin IS smart	1	-1	-1
UnCommon Without Category	orca whale HAVE sharp teeth	1	1	1
	orca whale HAVE black fins	1	1	1
	orca whale kill dolphin	1	1	1
Common & UnCommon Relationships Total				***3***

Best Example Score (4 * 2 + 3) 11

Example 15.4 - Multiplier Method Calculation

15.3.3.4. Multiplier & Combination Method

The *Multiplier & Combination Method* is a calculation that involves several algorithms to compute an entity's *best example* score. It is the same as the *Multiplier Method* except that it adds the number of the entity's emanating arcs to its base score before it is multiplied by the number of the category's primary parent's children. The following steps are performed by this method to calculate the *best example* score for each entity in the selected category:

- The number of primary children the category's primary parent has, is counted using the *Count Primary Children* algorithm.
- The entity's base score is calculated using the *Base Score Calculation.*
- The number of the entity's emanating arcs is added to its *best example* score.
- The entity's *best example* score is then multiplied by the number of its category's primary parent's children. This is done to normalize the entity's *best example* score for later processing, since the relationship numbers of other categories will be used in the computation of the entity's score.
- The number of relationships the entity has in common with the other entities in its category is added to the entity's score. This is done using the *Count Common Relationships* algorithm.
- The number of relationships the entity has in common with other entities in other the categories besides its own category is subtracted from the entity's *best example* score. This is done using the *Count Common Relationships* algorithm.
- The number of relationships the entity does not have in common with other entities in other the categories besides its own is added to the entity's score. This is done using the *Count UnCommon Relations* algorithm.

The *Multiplier & Combination Method* tries to determine if an entity's common and uncommon relationships within and without its category along with its emanating arcs can emulate a human's *best example* behavior.

For each of the entities to be ranked do:

- Calculate the entity's base score *(minimal method)*.
- Increment the entity's base score by the number of arcs the emanate from it.
- Count the entity's primary children.
- Set the entity's best example score equal to the entity's base score multiplied by the number of its primary children.
- Increment the entity's best example score by the number of relationships it has in common with other entities in its category.
- Decrement the entity's best example score by the number of relationships it has in common with other entities that are not in its category but are at a level below it.
- Increment the entity's best example score by the number of relationships it does **not** have in common with other entities that are not in its category but are at a level below it.

Algorithm 15.9 - Multiplier & Combination Method (summation)

An example of how the *Multiplier & Combination Method* works in calculating the *orca whale's best example* of a whale score is illustrated below. The semantic net displayed in figure 15.3 is used for input. To the left of each awarded point is the matching step used from the algorithm along with the corresponding relationship.

***Multiplier & Combination Method* Example**				
Rule	**Relationship / Rule Match**	**Quantity**	**Multiplier**	**Score**
Base Score + Emanating Arcs *(from example 15.3)*				***9***
Category's	mammal INCLUDES whale	1	1	1
Parent's Primary Children	mammal INCLUDES dolphin	1	1	1
Number of the Category's Parent's (mammal) Primary Children				***2***
	(Base Score + Emanating Arcs) × Category's Parent's Primary Children	9	2	***18***
Common Within Category	orca whale HAVE sharp teeth *matches with* toothed whale HAVE teeth	1	1	1
Common Without Category	orca whale IS smart *matches with* dolphin IS smart	1	-1	-1
UnCommon Without Category	orca whale HAVE sharp teeth	1	1	1
	orca whale HAVE black fins	1	1	1
	orca whale kill dolphin	1	1	1
Common & UnCommon Relationships Total				***3***
Best Example Score (9 * 2 + 3)				***21***

Example 15.5 - Multiplier & Combination Method Calculation

15.4. Machine Output

The machine displays the result of the *best example* ranking by showing an array of ranked entities in a vertical column. The entities are written in descending order according to their *best example* score while the rank of each entity is shown to its right. An example of a machine entity *best example* ranking is displayed below.

```
whale#0 Best Example Ranking
    orca whale#0 1
    toothed#0 whale#0 2
    blue whale#0 3
    sperm whale#0 4
    humpback whale#0 5
    toothless#0 whale#0 6
    largest#0 toothed#0 whale#0 7
```

```
    small#0 whale#0 8
    killer whale#0 9
```

Example 15.6 - Best Example Ranking Sample Output

The *best example* rankings by article are displayed in the following sub-sections. Each sub-section will be headed by the survey article used for input, followed by the corresponding ranking from each *best example* machine method.

15.4.1. Bear Best Example Ranking Machine Output

```
bear#0 Best Example Ranking Version 2
   black bear#0 1
   polar bear#0 2
   brown bear#0 3
   sun bear#0 4
   sloth bear#0 5
   mother#1 bear#0 5
   spectacled bear#0 5
   smallest#0 bear#0 8
   blue bear#0 8
   biggest#0 brown bear#0 10
   mother#1 sloth bear#0 10
   fiercest#0 brown bear#0 10
   moon bear#0 13
   grizzly bear#0 13
   kodiak bear#0 13
```

Figure 15.4 - Minimal Method Best Example Ranking (Bear Article)

```
bear#0 Best Example Ranking Version 3
   brown bear#0 1
   polar bear#0 1
   black bear#0 3
   spectacled bear#0 4
   sloth bear#0 4
   sun bear#0 4
   blue bear#0 4
   smallest#0 bear#0 8
   mother#1 bear#0 8
   kodiak bear#0 10
   mother#1 sloth bear#0 10
   biggest#0 brown bear#0 10
   moon bear#0 10
   grizzly bear#0 10
   fiercest#0 brown bear#0 10
```

Figure 15.5 - Combination Method Best Example Ranking (Bear Article)

```
bear#0 Best Example Ranking Version 4
   black bear#0 1
   polar bear#0 2
   brown bear#0 3
   sun bear#0 4
   mother#1 bear#0 5
   spectacled bear#0 5
   blue bear#0 7
   sloth bear#0 7
```

```
   smallest#0 bear#0 9
   mother#1 sloth bear#0 9
   biggest#0 brown bear#0 11
   moon bear#0 11
   grizzly bear#0 11
   fiercest#0 brown bear#0 11
   kodiak bear#0 11
```

Figure 15.6 - Multiplier Method Best Example Ranking (Bear Article)

```
bear#0 Best Example Ranking Version 5
   black bear#0 1
   polar bear#0 2
   brown bear#0 3
   sun bear#0 4
   mother#1 bear#0 5
   spectacled bear#0 6
   sloth bear#0 7
   mother#1 sloth bear#0 8
   smallest#0 bear#0 8
   blue bear#0 8
   biggest#0 brown bear#0 11
   fiercest#0 brown bear#0 11
   moon bear#0 13
   grizzly bear#0 13
   kodiak bear#0 13
```

Figure 15.7 - Multiplier & Combination Method Best Example Ranking (Bear Article)

15.4.2. Whale Best Example Ranking Machine Output

```
whale#0 Best Example Ranking Version 2
   orca whale#0 1
   toothed#0 whale#0 1
   blue whale#0 3
   sperm whale#0 4
   humpback whale#0 4
   toothless#0 whale#0 6
   largest#0 toothed#0 whale#0 7
   small#0 whale#0 8
   killer whale#0 9
```

Figure 15.8 - Minimal Method Best Example Ranking (Whale Article)

```
whale#0 Best Example Ranking Version 3
   toothed#0 whale#0 1
   blue whale#0 2
   humpback whale#0 2
   orca whale#0 4
   sperm whale#0 5
   toothless#0 whale#0 5
   largest#0 toothed#0 whale#0 7
   small#0 whale#0 7
   killer whale#0 9
```

Figure 15.9 - Combination Method Best Example Ranking (Whale Article)

```
whale#0 Best Example Ranking Version 4
```

```
   orca whale#0 1
   toothed#0 whale#0 2
   blue whale#0 3
   humpback whale#0 4
   sperm whale#0 5
   toothless#0 whale#0 6
   largest#0 toothed#0 whale#0 7
   small#0 whale#0 7
   killer whale#0 9
```

Figure 15.10 - Multiplier Method Best Example Ranking (Whale Article)

```
whale#0 Best Example Ranking Version 5
   orca whale#0 1
   toothed#0 whale#0 2
   blue whale#0 3
   sperm whale#0 4
   humpback whale#0 5
   toothless#0 whale#0 6
   largest#0 toothed#0 whale#0 7
   small#0 whale#0 8
   killer whale#0 9
```

Figure 15.11 - Multiplier & Combination Method Best Example Ranking (Whale Article)

15.4.3. Spider Best Example Ranking Machine Output

```
spider#0 Best Example Ranking Version 2
   wolf spider#0 1
   garden spider#0 2
   crab spider#0 3
   water spider#0 4
   trapdoor spider#0 5
   malmignette#0 6
   black widow#0 7
   mother#1 spider#0 7
   hammock spider#0 9
   zebra spider#0 9
   jumping spider#0 9
   house spider#0 9
   male#0 spider#0 13
   bird-eating spider#0 13
   funnel-web#0 15
   little#0 zebra spider#0 15
   female#1 malmignette#0 15
   American#0 black widow#0 18
```

Figure 15.12 - Minimal Method Best Example Ranking (Spider Article)

```
spider#0 Best Example Ranking Version 3
   wolf spider#0 1
   garden spider#0 2
   trapdoor spider#0 2
   malmignette#0 4
   black widow#0 4
   house spider#0 4
   hammock spider#0 4
```

```
   water spider#0 4
   jumping spider#0 4
   crab spider#0 4
   bird-eating spider#0 4
   male#0 spider#0 12
   mother#1 spider#0 12
   little#0 zebra spider#0 14
   zebra spider#0 14
   funnel-web#0 14
   American#0 black widow#0 14
   female#1 malmignette#0 14
```

Figure 15.13 - Combination Method Best Example Ranking (Spider Article)

```
spider#0 Best Example Ranking Version 4
   wolf spider#0 1
   garden spider#0 2
   water spider#0 3
   crab spider#0 3
   trapdoor spider#0 5
   black widow#0 6
   mother#1 spider#0 6
   malmignette#0 6
   hammock spider#0 9
   house spider#0 9
   zebra spider#0 11
   jumping spider#0 11
   male#0 spider#0 11
   bird-eating spider#0 11
   female#1 malmignette#0 15
   little#0 zebra spider#0 15
   funnel-web#0 17
   American#0 black widow#0 17
```

Figure 15.14 - Multiplier Method Best Example Ranking (Spider Article)

```
spider#0 Best Example Ranking Version 5
   garden spider#0 1
   wolf spider#0 2
   crab spider#0 3
   water spider#0 4
   trapdoor spider#0 5
   mother#1 spider#0 5
   malmignette#0 5
   black widow#0 8
   zebra spider#0 9
   house spider#0 10
   hammock spider#0 10
   jumping spider#0 12
   male#0 spider#0 12
   little#0 zebra spider#0 14
   female#1 malmignette#0 14
   bird-eating spider#0 14
   funnel-web#0 17
   American#0 black widow#0 18
```

Figure 15.15 - Multiplier & Combination Method Best Example Ranking (Spider Article)

15.5. Student Best Example Entity Survey

The medium used to validate the machine *best example* process was a survey given to a class of college students. In the survey, one question was asked to calibrate the student's *best example* ranking of seven entities within a given category. This was done after they had read a children's book on a particular animal category. Also, included with the survey was a definition of the term, "*Best Example*", that was located in the front. The format of the question is illustrated below.

*2) Rank the following objects according to what is the **Best Example** of an object from the previous article. Rank them in descending order. Give the Best Example object a "1" and the worst example a rank of "7". All objects must be ranked.* *___ entity A* *___ entity B* *___ entity C* *___ entity D* *___ entity E* *___ entity F* *___ entity G*

Example 15.7 - Best Example Question Format

The numerical result of the *best example* student survey was an array of seven integers for each student that answered the survey. The entries in the array had values between one and seven representing the student rank responses. For example, a student that answered question 2 displayed above had the following responses: entity A = rank of 3, entity B = rank of 1, entity C = 6, D = 4, E = 2, F = 7, and G = 5. This would be represented by the following column.

	Question 2 (best example)
Entity A	**3**
Entity B	**1**
Entity C	**6**
Entity D	**4**
Entity E	**2**
Entity F	**7**
Entity G	**5**

Example 15.8 - Student Survey Best Example Response Array

15.6. Validation of Machine Output

In the absence of a standardized testing instrument, measuring expert performance is difficult. Classically, human experts measure whether some new person is also an expert by how consistent the new person is with the recognized experts. A new person can be thought of as an expert if his performance is consistent with expert performance to the degree that the experts' performances are consistent with each other.

Two consistency scores were developed to evaluate the four machine *best example* methods.

- *Rank-Group-Consistency* score - Measures to what extent a machine method can rank a set of objects by *best example.*
- *Rank-Individual-Consistency* score - Determines which machine method performed the best at ranking a set of objects by *best example*.

15.6.1. Extent of Machine Best Example Ranking

Determining to what extent a machine can rank a set of objects by *best example*, the following steps were carried out:

- The *rank-group-consistency* score to be used to evaluate the machine's *best example* ranking performance was defined.
- The *rank-group-consistency* score formulation was explained.
- An example of how the *rank-group-consistency* score was computed and used was given.
- The *rank-group-consistency* score test was performed. This was done to determine to what extent a machine could rank set of objects by *best example*.

15.6.1.1. Rank-Group-Consistency Score Defined

The objective of the following *rank-group-consistency score* is to measure to what extent a machine method can rank a set of objects by *best example*. Determining this measure is done by comparing the machine's *rank-group-consistency score* to the *rank-group-consistency score* of the class. The *rank-group-consistency score* comparison is used to determine if the computer's rankings were similar to a class of students' rankings. This was done by comparing the computer's rankings to each student in the class and comparing each student's rankings with every other student's rankings in the class. The machine method's and class's consistency scores are then compared. This determines if the machine method performed "below", "the same as", or "higher than" the average student in ranking a set of objects by *best example*.

A *rank-group-consistency score* was developed for the machine. This was formulated by first, comparing the machine rankings to each student's rankings in the class. This resulted in an array of similarity scores. The similarity scores were then aggregated into one *rank-group-consistency score* by averaging them together. The idea is to discover how consistent the machine's rankings were with the rankings of the student aggregate. The *rank-group-consistency score* indicates how similar the machine was to the students aggregate in ranking a set of objects. A high average indicates a high level similarity.

A *rank-group-consistency score* was developed for the students as a group. This was done by first, comparing each student's rankings with every other student's rankings, excluding duplicate pairings. The similarity scores for the students were then aggregated into one *rank-group-consistency score* by averaging them together. The idea is to discover how generally consistent the students in the class were in their rankings amongst themselves.

The extent that a machine can rank a set of objects by *best example* can be measured by comparing the machine's *rank-group-consistency score* with the class's *rank-group-consistency score*. The scores were considered the same if they differed by no more than a tolerance value. In this experiment, the tolerance value was the variance of the class's *rank-group-consistency score*.

The similarity score between two sets of rankings was measured by calculating the Spearman Correlation Coefficient of the two. A correlation coefficient with a value close to one indicates a high level of similarity between the two. A value near zero indicates little or no similarity between the two. A value close to negative one indicates rankings that go in opposite directions. In chapter 14, it is shown how the *Spearman Rank Correlation Coefficient* is calculated and used in conjunction with the *rank-group-consistency* score.

15.6.1.3. Rank-Group-Consistency Score Example

To show how a machine method and student *rank-group-consistency* scores were calculated in the *best example* evaluation, an example is detailed below.

- First, five students are given an article about bears to read.
- Second, each student is asked to answer the following question.

  *2) Rank the following objects according to what is the **Best Example** of a bear from the previous article. Rank them in descending order. Give the Best Example of a bear a rank of "1" and the worst example of a bear a rank of "7". All objects must be ranked.*

  ___ *polar bear*
  ___ *brown bear*
  ___ *sun bear*
  ___ *kodiak bear*
  ___ *sloth bear*
  ___ *grizzly bear*
  ___ *black bear*

- The machine method also reads the same article about bears and answers the above question.
- The five students' and the machine's answers to the above ranking question are numerated and displayed in the table below.

Students/Machine	s1	s2	s3	s4	s5	machine
polar bear	2	1	3	2	2	2
brown bear	3	3	1	3	3	2
sun bear	4	7	7	6	4	4
kodiak bear	5	4	5	7	6	5
sloth bear	6	6	6	4	5	5
grizzly bear	7	5	4	5	7	7
black bear	1	2	2	1	1	1

Table 15.1 - Sample Student and Machine Best Example Rankings

- The above table can be interpreted as the following:

  Student s1 thought the best example of a bear was the black bear, then the polar bear, then brown bear, etc..

  Student s2 thought the best example of a bear was the polar bear, black bear, brown bear, kodiak bear, etc..

  The machine indicated the best example of a bear was the black bear, then the polar bear, then the brown bear, then the sun bear, and etc..

- Next, the machine is correlated with all five students and the students are correlated with each other, excluding any duplicate pairings, to compute their similarity scores. The Spearman's Rank

Correlation Coefficient formula was used in calculating the correlations. The result of these correlations is displayed in the table below:

	Machine	Students				
Students	**m**	**s1**	**s2**	**s3**	**s4**	**s5**
s1	0.95					
s2	0.67	0.71				
s3	0.59	0.57	0.82			
s4	0.87	0.71	0.71	0.71		
s5	0.99	0.96	0.64	0.54	0.82	

Table 15.2 - Best Example Similarity Scores Example

- The machine *rank-group-consistency score* is computed by averaging the correlation coefficients in column "m" (second column). The class's *rank-group-consistency score* is computed by averaging the correlation coefficients in columns "s1" through "s5". The tolerance value for the class's *rank-group-consistency score* is calculated by taking the variance of columns "s1" through "s5". The results of these three calculations are displayed below.

	Machine	Students
Mean	0.81	0.72
Variance		0.02

Table 15.3 - Best Example Rank-Group-Consistency Scores Example

- Finally, after the *rank-group-consistency score*s have been calculated they are then used to evaluate how well the machine could rank a set of entities by *best example*. In this example, the machine method has exceeded the students score by more than the tolerance. Therefor it was able to rank entities by *best example* better than the average student.
- *Rank-Group-Consistency* score for the class of students indicates how the well the average student did ranking a set of objects by *best example*. The table above tells us how well the machine method did against the average student. The machine method can do worse, the same, or better than the average student.

15.6.1.4. Rank-Group-Consistency Score Test

The *rank-group-consistency* scores are used to determine to what extent a machine method can rank a set of objects by *best example*. This is done by comparing the class of students' *rank-group-consistency* score with that of the four *best example* machine methods' scores. To make the consistency score comparisons easy to interpret, they were put into a table. The table holds the *rank-group-consistency* scores for the survey question, "*Rank the set of objects by best example*".

A description of the columns and rows of the *rank-group-consistency* score table is detailed here. The "Question" row refers to what survey question was asked. The "Class Score" row displays the class's *rank-group-consistency* scores for the three surveys that were distributed to the class of college students. The "Tolerance Value" row displays the class's *rank-group-consistency* score variances. The "Machine Methods" heading indicates the start of the four machine method's consistency scores. The *Minimal Method, Combination Method, Multiplier Method,* and *Multiplier & Combination* rows display each of the four machine method *rank-group-consistency* scores.

Best Example Rank-Group-Consistency Scores

Question	*Rank by Best Example*		
Article	**Bears**	**Whales**	**Spiders**
Class Score	0.57	0.15	0.31
± **Tolerance Value**	0.07	0.19	0.12
Machine Methods			
Minimal Method	0.62	0.40	0.08
Combination Method	0.60	0.33	0.29
Multiplier Method	0.62	0.41	0.05
Multiplier & Combination	0.62	0.41	0.15

Table 15.4 - Best Example Rank-Group-Consistency Score Test

The criterion for this performance evaluation is the machine method's rank consistency with the students compared to the rank consistency among the students themselves. The class consistency score reflects what the average student's *rank-group-consistency* score was. The class's consistency score is the consistency score of each student pairing in the class, excluding any duplicates, averaged together into one score. If the machine method is said to have performed as well as the average student, this means the method was just as consistent as the class of students was in ranking objects by *best example*.

The *Minimal Method* performance was inconclusive in terms of the average student in ranking a set of objects by *best example*. This is attested to different performance levels for each of the surveys. In the bear survey the *Minimal Method* had the same *rank-group-consistency* score as the class of students did. This meant that the method performed as well as the average student. In the whale survey the *Minimal Method* had a higher *rank-group-consistency* score than the class of students did. This meant that the method performed better than the average student. In the spider survey the *Minimal Method* had a lower *rank-group-consistency* score than the class of students did. Meaning that the method performed worse than the average student did. With three different levels of performance, it is inconclusive to what extent the *Minimal Method* can rank a set of objects by *best example.*

The *Combination Method* performed just as well as the average student in ranking a set of objects by *best example*. The *Combination Method* had the same *rank-group-consistency* score as the class did in all three surveys. A method's consistency score is considered the same if it does not exceed the class's score by more than the tolerance value.

The *Multiplier Method* performance was inconclusive in terms of the average student in ranking a set of objects by *best example*. This is attested to different performance levels for all three surveys. In the bear survey the *Multiplier Method* performed as well as the average student did. In the whale survey the *Multiplier Method* had a higher *rank-group-consistency* score than the class of students did. This meant that the method performed better than the average student did. In the spider survey the *Multiplier Method* performed worse than the average student did. With three different levels of performance, it is inconclusive to what extent the *Multiplier Method* can rank a set of objects by *best example*.

The *Multiplier & Combination* method performance was inconclusive in terms of the average student in ranking a set of objects by *best example*. This is attested to different performance levels for each of the surveys. In the bear survey the *Multiplier & Combination* method had the same *rank-group-consistency* score as the class of students did. This meant that the method performed as well as the average student did. In the whale survey the *Multiplier & Combination* method performed better than the average student did. In the spider survey the *Multiplier & Combination* method had a lower *rank-group-consistency* score than the class of students did. This meant that the method performed worse than the average student did. With three different levels of performance, it is inconclusive to what extent the *Multiplier & Combination* method can rank a set of objects by *best example.*

15.6.2. Finding the Best Machine Method

Determining which machine method can rank a set of objects by *best example* the best, the following steps were carried out:

- The *rank-individual-consistency* score was defined.
- An example of how the *rank-individual-consistency* score was calculated and used.
- The *rank-individual-consistency* score test was performed. This was done to determine which machine method was the best at ranking a set of objects by *best example*.

15.6.2.1. Rank-Individual-Consistency Score Defined

The objective of the *rank-individual-consistency score* is to determine which machine method is the most accurate at ranking a set of objects by *best example*. One way of determining this measure is by comparing the machine's *rank-individual-consistency score* to the *rank-individual-consistency score* of each student in the class. The *rank-individual-consistency score* is used to determine if the computer's rankings were similar to a particular student's rankings. The results of the comparisons are then tabulated as to the number of students that the machine method did "worse than", "the same as" or "better than" in ranking a set of objects. A qualitative measure is then applied to the counts for each method to determine which method performed the best in terms of student rank consistency.

A *rank-individual-consistency score* was developed for the machine and each of the students in the class. This was done by comparing the machine rankings to each student's rankings excluding the student the machine was being compared with. The idea is to discover how consistent the machine's rankings and the student in question were with the rankings of the rest of the class. One such score was developed for each student in the class other than the one being compared to the machine. These scores were aggregated by averaging them together. There is one such average for each machine/student comparison. The same thing was done for all students in the class to see how internally consistent the students were in their rankings. These averages indicate how similar the machine was to the students in ranking objects. A high average indicated a high level similarity.

The *rank-individual-consistency score*s of all machine/student and student/student comparisons were aggregated by counting how many times the machine was "better than", "the same as", and "worse than" the students. The machine and student *rank-individual-consistency score*s were considered the same if they differed by no more than a tolerance value. In this experiment, the tolerance value was the variance of the student's *rank-individual-consistency score*.

The similarity between two sets of rankings can be measured by calculating the Spearman Correlation Coefficient of the two. The Spearman Correlation Coefficient was used, because it is specifically constructed to correlate two pairs of rankings. A correlation coefficient with a value close to one indicates a high level of similarity between two rankings. A value near zero indicates little or no similarity between two rankings. A value close to negative one indicates rankings that go in opposite directions.

The *rank-individual-consistency* score is used in a different way than the *rank-group-consistency* score. The difference is that the *rank-individual-consistency* score measures the machine method's performance by comparing the method's performance to each student's performance. On the other hand the *rank-group-consistency* score measures how the method did compared to the entire class. When the *rank-individual-consistency* score is measured against each student's score, counts are accumulated. These counts indicate if the machine method performed "worse than", "the same as", or "better than" the selected student in ranking a set of objects. The criterion for measuring performance is who was more consistent with the rest of class in ranking a set of objects, the machine method or the student. The counts are interpreted using a qualitative measure to determine which machine method performed the best in ranking a set of objects.

15.6.2.2. Qualitative Measure Defined

A Qualitative measure was needed to aggregate the machine method's three performances into one score. The measure is used to determine which machine method performed the best at ranking a set of objects by *best example*. The machine method with the highest qualitative measure would be considered the best method at ranking a set of objects. The criterion for this qualitative measure is based upon the most consistency with student *best example* rankings.

The qualitative measure must take into account when the machine does "worse than", "the same as" or "better than" the students. This was done by awarding points to the machine method according to how well it measured up to student's performance. If the machine method did worse than the student, no points were awarded to its tally. Every time the machine method performed the same as the student, a point was awarded to its tally. Every time the machine method performed better than the student, two points were awarded to its tally. The tallies were then accumulated into a total for each survey. The total was then divided by the total number of possible points a machine method could earn for each survey, to compute a sub score. Finally, when all the sub scores had been computed, they were averaged together to calculate the method's qualitative measure.

The qualitative measure to determine which machine method did the best in ranking a set of objects by *best example* is calculated for each method using the following algorithm.

1. *For each survey, perform the following steps.*
   - 1.1 *Multiply the number of times the method performed worse than the students by zero.*
   - 1.2 *Multiply the number of times the method performed the same as the students by one.*
   - 1.3 *Multiply the number of times the method performed better than the students by two.*
   - 1.4 *Add the points calculated in the three previous three steps together.*
   - 1.5 *Divide the total from the previous step by the number of students who participated in the survey multiplied by two. This measure is called the method's survey sub score.*
2. *The method's qualitative score is computed by averaging its three survey sub scores together.*

15.6.2.3. Rank-Individual-Consistency Score Example

To show how the machine method and student *rank-individual-consistency* scores were calculated, a detailed example is given below. The example displays how the *rank-individual-consistency* score is used to determine which machine method performed the best in ranking a set of objects by *best example*. In the first several steps, only one machine method's consistency score is demonstrated to avoid duplicate effort.

- First, five students are given an article about bears to read.
- Second, each student is asked to answer the following question.

  *2) Rank the following objects according to what is the **Best Example** of a bear from the previous article. Rank them in descending order. Give the Best Example of a bear a rank of "1" and the worst example of a bear a rank of "7". All objects must be ranked.*
  ___ polar bear
  ___ brown bear
  ___ sun bear
  ___ kodiak bear
  ___ sloth bear
  ___ grizzly bear
  ___ black bear
- The machine method also reads the same article and answers the above question.
- The five students' and the machine's answers to the above ranking question are numerated and are displayed in the table below.

Students/Machine	s1	s2	s3	s4	s5	machine
polar bear	2	1	3	2	2	2
brown bear	3	3	1	3	3	2
sun bear	4	7	7	6	4	4
kodiak bear	5	4	5	7	6	5
sloth bear	6	6	6	4	5	5
grizzly bear	7	5	4	5	7	7
black bear	1	2	2	1	1	1

Table 15.5 - Sample Student and Machine Best Example Rankings

- The above table can be interpreted as the following:

  Student s1 thought the best example of a bear was the black bear, then the polar bear, then the brown bear, etc..

  Student s2 thought the best example of a bear was the polar bear, then the black bear, then the brown bear, then the kodiak bear, etc..

  The machine indicated the best example of a bear was the black bear, then the polar bear and then the brown bear, then the sun bear, etc..

- For each of the students (*s1 through s5*) , that were be compared with the machine, the following was done:

  - The machine rankings were correlated, using the Spearman Correlation Coefficient, with each of the students excluding the student compared with.

- The student that is to be compared with the machine has his rankings correlated, using the Spearman Correlation Coefficient, with all the other participating students.

- The numerical result of the above action is then recorded in the table below.

	Machine Correlations					Student Correlations				
Method/Student	**m1/s1**	**m1/s2**	**m1/s3**	**m1/s4**	**m1/s5**	**s1**	**s2**	**s3**	**s4**	**s5**
s1		0.95	0.95	0.95	0.95		0.71	0.57	0.71	0.96
s2	0.67		0.67	0.67	0.67	0.71		0.82	0.71	0.64
s3	0.59	0.59		0.59	0.59	0.57	0.82		0.71	0.54
s4	0.87	0.87	0.87		0.87	0.71	0.71	0.71		0.82
s5	0.99	0.99	0.99	0.99		0.96	0.64	0.54	0.82	
***rank-individual-consistency* score (average)**	**0.78**	**0.85**	**0.87**	**0.80**	**0.77**	**0.74**	**0.72**	**0.66**	**0.74**	**0.74**
tolerance value (variance)						0.03	0.01	0.02	0.00	0.04

Table 15.6 - Sample Machine and Student Correlations for the Rank-Individual-Consistency Score

- The *rank-individual-consistency score*s are then calculated for each of the machine and student runs, columns "m/s1" through "m/s2" for the machine and columns "s1" through "s5" for the students, by taking the average of each column. The consistency score in column "m/s1" is the machine score excluding the correlation between the student "s1". This is the score that will be compared to student "s1's" score.

- The tolerance values were derived from the variance of each student's *rank-individual-consistency score*.

- The four machine *rank-individual-consistency score*s are then compared to their corresponding student *rank-individual-consistency score*s (i.e., *machine score m/s1 is compared to student score s1*). These comparison pairings are displayed in the table below. Notice that the machine score from the above table in column "m/s1" was positioned in column "s1" in the below table. The scores were considered the same if they differed by no more than the corresponding tolerance value.

Rank-Individual-Consistency Scores	**s1**	**s2**	**s3**	**s4**	**s5**
student	0.74	0.72	0.66	0.74	0.74
tolerance value	0.03	0.01	0.02	0.00	0.04
machine method	0.78	0.85	0.87	0.80	0.77

Table 15.7 - Sample Machine and Student Rank-Individual-Consistency Score Comparisons

- The machine did better than students' "s1", "s2", "s3" and "s4" in ranking a set of objects by *best example.* The performance criterion was rank similarity with the rest of the participating students excluding the student the machine was compared with. The machine did as well as student "s5" in ranking a set of objects by *best example.*

- To demonstrate how one machine method is determined to be better than another, a second machine method's *rank-individual-consistency* scores are introduced. The machine method whose consistency score was just calculated will be referred to as *method one*. The new machine method being introduced will be referred to as *method two*.

Rank-Individual-Consistency Scores	**s 1**	**s 2**	**s 3**	**s 4**	**s 5**
student	0.74	0.72	0.66	0.74	0.74
tolerance value	0.03	0.01	0.02	0.00	0.04
machine method one	0.78	0.85	0.87	0.80	0.77
machine method two	0.81	0.71	0.77	0.73	0.72

Table 15.8 - Sample Machine and Student Rank-individual-consistency score Comparisons

- Using the above table, the number of times a machine method performed "worse than", "the same as", or "better than" the students is accumulated and displayed in the table below.

Consistency Score Student Comparison Counts			
Question	*Rank the objects by best example*		
Article	**Bears**		
Number of times method did worse, the same, or better than the students.	**worse**	**same**	**better**
Machine Method			
Machine Method One	0	0	5
Machine Method Two	1	2	2

Table 15.9 - Sample Consistency Score Student Comparison Counts

- The table above indicates that the machine *method one* did better than all five students in this example, in ranking a set of objects by *best example*. *Method two* on the other hand, performed worse than one student, the same as two other students, and better than two students. The criterion for performance is rank consistency with all the other participating students.
- To determine which machine method performed the best the qualitative measure is applied. The calculations for the qualitative measure are displayed in the table below.

Best Example Ranking Qualitative Measure Calculations					
Question	*Rank the objects by best example*				
Article	**Bears**				
Number of times method did worse, the same, or better than the students.	**worse**	**same**	**better**	**tally**	**Qualitative Measure**
Machine Method					
Machine Method One	0 x 0	0 x 1	5 x 2	10	10 / 10 = 1.0
Machine Method Two	1 x 0	2 x 1	2 x 2	6	06 / 10 = 0.6

Table 15.10 - Sample Best Example Ranking Qualitative Measure Calculations

- Observing the qualitative measures pictured in the table above. It can be seen that machine *method one* performed better than machine *method two*. This can be attested, since *method one's* score of 1.0 is higher than *method two's* score of 0.6. This indicates that *method one* is better than *method two* in ranking a set of objects by *best example*.

15.6.2.4. Rank-Individual-Consistency Score Test

The *rank-individual-consistency* scores in conjunction with the qualitative measure, determine which machine method can rank a set of objects the best. The results of the ranking experiment are calculated and elaborated upon down below.

A description of the columns and rows of the table used in evaluating the machine methods is detailed here. The "Question" row refers to the survey question that was asked. The "Article" row refers to the survey that was used. The "Machine Method" heading indicates the start of machine method results. The "worse" column indicates the number of students a machine method had a lower *rank-individual-consistency* score than. The "same" column indicates the number of students a machine method had the same consistency score than. The "better" column indicates the number of students a machine method had a higher consistency score than. The column labeled with "Qualitative Score" displays each machine method's qualitative score. The rows *Minimal Method, Combination Method, Multiplier Method,* and *Multiplier & Combination* display the accumulated counts for each machine method.

***Rank-Individual-Consistency* Score Count Comparisons**										
Question	*Rank the objects by best example*									
Article	**Bears**			**Whales & Dolphins**			**Spiders**			
Number of times method did worse, the same, or better than the students.	**worse**	**same**	**better**	**worse**	**same**	**better**	**worse**	**same**	**better**	**Qualita -tive Score**
Machine Method										
Minimal Method	2	6	2	0	4	7	14	6	0	**0.49**
Combination Method	4	4	2	0	8	3	6	8	6	**0.51**
Multiplier Method	2	6	2	0	4	7	17	3	0	**0.46**
Multiplier & Combination	2	6	2	0	5	6	13	7	0	**0.48**

Table 15.11 - Rank-Individual-Consistency Count for Best Example Rankings

The *Combination Method* was the best machine method in ranking objects by *best example*. This was concluded since the *Combination Method's* qualitative score was the highest when compared to other method's scores. Meaning that the *Combination Method* had the least number of students who performed better than it and the most number of students that performed the same or worse than it. Performance was based upon consistency with the rest of the class in ranking a set of objects by *best example*.

The *Multiplier Method* was the worst machine method in ranking objects by *best example*. This was attested since the *Multiplier Method's* qualitative score was the lowest when compared to other method's scores. Meaning that the *Multiplier Method* had the most number of students who performed better than it and the least number of students that performed the same or worse than it.

15.6.3. Investigator's View of the Machine's Best Example Rankings

Output of the machine *best example* ranking method looked good. Looking at the machine's *best example* rankings, it seemed the rankings made sense after reading the natural language text, with one exception. For example, "brown bear", "polar bear" and "black bear" were the bears mentioned most often and came in at the top bear rankings. This was the same for the spider article where the "wolf" and "garden" spiders got high rankings. The exception to the good rankings happened in the processing of the whale article with the "toothed whale" receiving the highest *best example* ranking. On the other hand, it was very positive to see the "blue whale" and the "humpback whale" come in

second in the rankings. Another good observation about the machine's *best example* rankings was that it always gave out at least five different rankings.

There were two things that could make the *best example* ranking method better. The first thing that could be improved is the differential in rankings between animal subsets and animal sub species. At times, the *best example* method ranked species subsets along with or higher than the sub species. For example, the machine method ranked the "toothed whale" higher than any other whale when the whale article was processed. When humans think of best examples they usually think of sub species, not a subset of the species. This problem is also related to the one experienced by the machine in the importance rankings. Possible solutions to this sub specie and specie subset ranking problem would be to treat the sub species and specie subsets differently and process a higher volume of natural language text.

The second thing that would be nice to see in the *best example* rankings is a higher disparity in the rankings. In other words, the *best example* method should use more ranks when ranking objects by *best example*. This is especially true in the bear rankings where the rank of "10" was given out six times and in the spider rankings where the rank of "4" was given out eight times. A possible solution to this rank disparity problem is to process a higher volume of natural language text. This would provide more information to differentiate between the different ranked objects.

15.7. Conclusion

To show that a machine has the conceptualization ability to rank related entities as to which is the *best example* of a related category was a complex process. First, different machine methods had to be developed to rank entities by *best example* criteria. Second, tools had to be found that could evaluate the results of the machine methods. Next, students had to be surveyed, so there was data to gauge the accuracy of the machine methods. Finally, the results of the evaluations were collected and summarized to determine to what extent a machine method could rank related entities by *best example* and which machine method could do this the best.

The four different machine methods that were developed to rank entities by *best example* criteria were:

- *Minimal Method* - This method computes an entity's *best example* score using three criteria. First, the method measures the strength of the relation between the scoring entity and its category entity node and initializes the entity's score with the appropriate number of points. Next, the method increments an entity's *best example* score by the number of relationships the entity has in common with the other entities within its category. Last, the method decrements an entity's *best example* score by the number of relationships the entity has in common with other entities within its category where the relations have opposite polarity.
- *Combination Method* - This method computes an entity's *best example* score by using the result of the *Minimal Method* plus adding the number of arcs that emanate from the scoring entity.
- *Multiplier Method* - This method computes an entity's *best example* score by first initializing the entity's score with the result of the *Minimal Method*. Next, the entity's score is multiplied by the number of primary children its category's primary parent has. Then the method either adds or subtracts points from the entity's score using the criteria of common and uncommon relationships within and without the scoring entity's category.
- *Multiplier & Combination Method* - This method computes an entity's *best example* score in four steps. First the entity's *best example* score is initialized with the result of the *Minimal Method*. Second, the score is then incremented by the number of arcs that emanate from the scoring entity. Next, the entity's score is multiplied by the number of primary children its category's primary

parent has. Last, the method either adds or subtracts points from the entity's score using the criteria of common and uncommon relationships within and without the scoring entity's category.

A *rank-group-consistency score* was developed to evaluate the machine *best example* ranking methods. The score essentially computes the similarity of machine and human rankings. The score was computed for both the machine and the class of students. The machine's consistency *score* entails correlating the machine *best example* rankings with each student's rankings and averaging the coefficients into one score. The class's consistency score involves correlating each student's *best example* rankings with every other student's rankings, excluding duplicated pairings, and then averaging the coefficients into one score. A tolerance value was calculated by taking the variance of the class's correlation coefficients. The extent that a machine can rank a set of objects by *best example* was measured by comparing the machine's *rank-group-consistency score* with the class's *rank-group-consistency score*. The scores were considered the same if they differed by no more than the tolerance value.

The *rank-individual-consistency* score was developed to determine which machine method could rank a set of objects by *best example* the best. The *rank-individual-consistency* score is used in a different way than the *rank-group-consistency* score. The difference is that the *rank-individual-consistency* score measures the machine method's performance by comparing the method's performance to each student's performance, while the *rank-group-consistency* score measures how the method did compared to the entire class. When the *rank-individual-consistency* score is measured against each student's score, counts are accumulated. These counts indicate if the machine method performed "worse than", "the same as", or "better than" the students in ranking a set of objects by *best example*. The criterion for measuring performance is who was more consistent with the rest of class in ranking a set of objects, the machine method or the student. A qualitative measure is calculated using the counts. This qualitative measure determines which machine method performed the best in ranking a set of objects by *best example*.

Three surveys were taken from a college student population to evaluate the performance of the machine's *best example* methods. In each survey the students were handed a packet that consisted of relevant definitions, an article, and a set of questions. The relevant definitions supplied each student with a standard definition of what is a *best example* of a category and an example of that concept. The article received by the students was about a specific entity(s) that they were to read and to use as a reference in answering questions. The question the students were given, "*Rank the following objects according to what is the Best Example of an object from the previous article*," was used to capture their *best example* rank response. The data derived from the student responses to the survey was then used to validate the results of the machine's methods.

The *rank-individual-consistency* score in conjunction with a qualitative measure determined which machine method performed the best at ranking a set of objects by *best example*. In this experiment the *rank-individual-consistency* score indicated that the *Combination Method* was the best, at ranking a set of objects by *best example*. The criterion for this evaluation was student rank consistency. This was due to the fact that this machine method performed worse than least number of numbers and better than the greatest number of students.

Determining to what extent a machine can rank a set of objects by *best example* was a twofold process. First, the best performing machine method had to be selected. This was done using the *rank-individual-consistency* score and it picked the *Combination Method*. Second, the *Combination Method's* performance was evaluated in terms of the average student. In this experiment using the *rank-group-consistency* score as a gauge the following was indicated. The *Combination Method* could rank a set of objects by *best example* as well as the average student. The average student is defined as a student who participated in the survey and was compared with each machine method and his fellow students in ranking objects by *best example*. If any of the other machine methods would have been

selected as the best method to rank a set of objects by *best example*, then the extent that a machine could rank a set of objects by *best example* would have been inconclusive.

16. Basic Level Entities

16.1. Introduction

Basic level entities are entities: that have an integrity of their own, that take on the earliest forms of categorization, that are prototypical, that are cognitively simply, that capture bodily experiences, that captivate our imagination, that are the easiest for children to learn, that are highly specified categories, that contain much information, that are more fundamental than other levels of classifications, that are based upon motor actions, and that envision physical attributes. *Basic level* entities as stated, play a large role in the human conceptualization process. If a machine method has the ability to distinguish *basic level* entities from other entities, then a good measure of a machine's conceptualization facilities would be determined by how well it can select *basic level* entities from a set of entities. Many experts and scientists claim that *basic level* entities are an important facet of conceptualization process for the following reasons:

- Lackoff (1987) in his book "Women, Fire, and Dangerous Things" says that *basic level* entities are cognitively simple, easy to process by humans, simple to use, are prototypical, and have an integrity of their own. [p. 49, 58, 199, 267-268]
- Lackoff also explicates that *basic level* entities embody our environment:

  > "*Basic level* category structure reflects the bodily nature of the people doing the categorizing, since it depends on gestalt perception and motor movements..." [Lackoff, 1987, p. 371]

- Gelman claims that *basic level* entities are the easiest to learn:

  > "*Basic level* categories are especially easy for children to learn; superordinates are especially difficult (Markman & Callanan, 1984). Accordingly, the *basic level* may have special status for promoting children's inferences. ... Second graders also relied on category level and property generalizability to guide induction. They, too, drew fewer inferences to superordinate or unrelated than *basic level* categories." [Gelman, 1988, p. 68, 78]

- Rosch and Mervis explicate that *basic level* entities are specific and embody much material:

  > "Categorizations which humans make of the concrete world are not arbitrary but highly determined. In taxonomies of concrete objects, there is one level of abstraction at which the most basic category cuts are made. Basic categories are those which carry the most information, possess the highest category cue validity, and are, thus, the most differentiated from one another. ... basic objects are the most inclusive categories whose members: (a) possess significant numbers of attributes in common, (b) have motor programs which are similar to one another, (c) have similar shapes, and (d) can be identified from averaged shapes of members of the class." [Rosch, Mervis, 1976, p. 382]

- Mervis embellishes on the idea that *basic level* categories are more fundamental, easier to learn, and have similar attributes within themselves but not without:

  > "Children's initial categories are basic-level categories. This seems reasonable, because ... , categories at the *basic level* are more fundamental psychologically than categories at other taxonomic levels. For example, chair (basic-level category) is more fundamental than either kitchen chair (a subordinate category) or furniture (a superordinate category). Categories at the *basic level* 'stand out' as categories. Theses categories are based on large clusters of (subjectively) correlated attributes

that overlap very little from category to category. In our world, these basic-level categories are the most general categories whose members share similar overall shapes (or similar parts in particular configurations; and similar functions or characteristic actions." [Mervis, 1987, p. 202]

- Berlin, Breedlove, and Raven discuss a folk-lore category level that is equivalent to our *basic level* categories among the Tzeltal peoples. This level is based upon motor actions, physical appearance, imagination, and cultural significance. [Berlin, Breedlove, Raven, 1974, pp. 25-45]

16.2. Purpose

The purpose of the *Basic Level Entities* section is to demonstrate the degree of similarity between machine and human entity *basic level* selection. This process is developed by:

- Illustrating and elaborating the process involved in choosing a *basic level* entity
- Explaining the machine method's *basic level* output
- Discussing the student survey's *basic level* question
- Explicating the measures used to evaluate the machine method's performance
- Validating the machine method's output using the student survey results

With the previous steps executed, the quality and human similarity of machine *basic level* selections can be evaluated to some degree.

16.3. Process

The approach taken to determine if an entity is a *basic level* entity was based upon the amount of knowledge known about that entity. Where knowledge is equivalent to the number of relations that emanate from an entity and its primary ancestors. *Basic level* entities are the ones whose knowledge contained within them is greater than all their primary descendants and ancestors. The process and algorithms to be discussed follow this logic.

The method to determine what entities are *basic level* ones requires two steps to execute. The first step, uses a recursive algorithm that calculates the *basic level* score for each primary child in the semantic net starting at the root. The second step, performs another recursive algorithm that starts at the root of the net and prints out the *basic level* entities that meet a set of output restrictions. The *basic level* score calculation and restrictive print steps are explicated in detail below.

16.3.1. Basic Level Score Calculation

The *basic level* score calculation uses one algorithm, *Calculate Basic Score*, that is called recursively down the semantic net starting at the root. The algorithm first finds the entity's primary parent. It then computes the *basic level* score by adding together the entity's number of emanating arcs and the number of its parent's *non-inclusive* arcs. *Non-inclusive* arcs are all the arcs excluding the ones labeled with "INCLUDES," "HAS SUBSET," and "INSTANCE." Next, the algorithm proceeds to increment a variable, that it is to be passed down to the entity's primary children, by the number of *non-inclusive* arcs emanating from the entity. Finally, the algorithm is called by the entity's primary children. The *Calculate Basic Score* algorithm is detailed below.

For each entity starting at the root of the net do:

- Accumulate the entity's number of emanating arcs.
- Add to the above total, the entity's primary parent's number of non-inclusive arcs.
- Call the algorithm for all the entity's primary children, passing along the entity's number of non-inclusive arcs.

Algorithm 16.1a - Calculate Basic Score (summation)

```
basic_score    /*Basic Level Score*/
pass_score     /*Stores the number of non-membership arcs
entity         /*Current entity*/
distance       /*Distance away from the root of the semantic net*/
arcs_out       /*The number of arcs that emanate from an entity*/
relation[]     /*Arcs emanating from the current entity*/
children[]     /*Current entity's children connected by relation*/

/*Calculate Basic Score Algorithm*/
pass_score = 0
If entity.distance > 0 Then
    i = 0
    /*Find entity's parent*/
    While i < entity.arcs_out Do
        If (entity.relation[i] = "ISA" Or "INSTANCE OF"
            Or "SUBSET OF") Then
            Break
        End If
        i = i + 1
    End While
    /*Calculate entity's basic level score*/
    entity.basic_score =  entity.children[i].pass_score +
                    entity.arcs_out
    /*Count the number of non-membership relations*/
    i = 0
    While i < entity.arcs_out Do
        If (entity.relation[i] = "ISA" Or "INSTANCE OF"
            Or "SUBSET OF") Then
            Continue
        Else
            pass_score = pass_score + 1
        End If
        i = i + 1
    End While
End If
/*Recursively call the algorithm down the net*/
i = 0
While i < entity.arcs_out Do
    If (entity.relation[i] = "ISA" Or "INSTANCE OF" Or "SUBSET OF")
Then
        entity.children[i].CalculateBasicScore()
    End If
    i = i + 1
End While
```

Algorithm 16.1b - Calculate Basic Score (technical)

Figure 16.1 - Sample Semantic Net

An example is given below, showing how the *Calculate Basic Score* algorithm works using the diagram of a semantic net illustrated above. It demonstrates the *basic level* score calculations for the mammal, dolphin and toothless dolphin entities.

Basic Level Score Calculation Example		
Entity	**Rule**	**Result**
mammal	Parent is *root entity*	
mammal	Number of emanating arcs	4
root entity	Number of *non-inclusive* arcs	0
mammal	**Basic Level Score**	**4**
dolphin	Parent is *mammal*	
dolphin	Number of emanating arcs	3
mammal	Number of *non-inclusive* arcs	2
dolphin	**Basic Level Score**	**5**
toothless dolphin	Parent is *dolphin*	
toothless dolphin	Number of emanating arcs	1
dolphin	Number of *non-inclusive* arcs	2
toothless dolphin	**Basic Level Score**	**3**

Example 16.1 - Basic Level Score Calculations

16.3.2. Restrictive Print

The *Restrictive Print* algorithm determines which entities are the *basic level* entities. This algorithm starts at the root of the semantic net and is passed recursively down to each of the entity's primary children excluding instances and subsets. The algorithm first compares the entity's *basic level* score with that of its ancestor's. If the entity's score is greater than its ancestor's then the entity goes through further checks. Otherwise, it passes the algorithm along to its primary children. Next, the algorithm compares the entity's *basic level* score with that of its primary descendants'. If the entity's score is greater than its primary descendant's then the entity goes through one more check. Otherwise, it passes the algorithm along to its primary children. Finally, the entity is checked to see if it has any primary children or subsets. If the entity does have primary children or subsets, then the entity is printed out as a *basic level* entity and processing is returned to the calling procedure. Otherwise, it passes the algorithm along to its primary children. The *Restrictive Print* algorithm is displayed below.

```
Starting at the root of the semantic net do:
    If the entity's score is greater than its ancestors
        If the entity's score is greater than its primary descendants
            If the entity has primary children or subsets
                Select the entity has a basic level object.
            Else
                Pass the algorithm to the entity's primary children.
        Else
```

Pass algorithm to the entity's primary children.

Else

Pass algorithm to the entity's primary children.

Algorithm 16.2a - Restrictive Print (summation)

```
basic_score       /*Basic Level Score*/
ancestors_score   /*Entity's primary ancestor's basic level score
descendants_score /*Entity's primary descendant's basic level score
entity            /*Current entity*/
distance          /*Distance away from root of the semantic net*/
arcs_out          /*The number of arcs that emanate from an entity*/
relation[]        /*Arcs emanating from the current entity*/
children[]        /*Entity's children connected by relation*/

/*Restrictive Print Algorithm*/
RestrictivePrint(ancestors_score)
Begin
    If (entity.basic_score > ancestors_score) Then
        If (entity.basic_score > descendants_score) Then
            i = 0
            flag = False
            While i < entitiy.arcs_out Do
                If (entity.relation[i] = "INCLUDES" Or "HAS SUBSET")
                Then
                    Flag = True
                    Break out of loop
                End If
                i = i + 1
            End While
            If (flag = True) Then
                Print entity
                Return
            End If
        End If
    End If
    /*Make More Descendants Eligible For Basic Level Printing*/
    If (entity.basic_score > ancestors_score) Then
        ancestors_score = entity.basic_score
    End If
    /*Recursively Call Algorithm Down The Net*/
    i = 0
    While i < entity.arcs_out Do
        If (entity.relation[i] = "INCLUDES") Then
            entity.children[i].RestrictivePrint(entity.basic_score)
        End If
        i = i + 1
    End While
End
```

Algorithm 16.2b - Restrictive Print (technical)

A discussion is elaborated upon here, to explicate how the *Restrictive Print* algorithm decides which entities are *basic level* ones using the values obtained in example 16.1. Listed below are four entities, that are contained in a primary membership path, along with their reasons why they were selected or not selected as *basic level* entities:

- *Root Entity* - This entity is excluded from *basic level* membership, since it is the root of the semantic net and it has a zero distance value.
- *Mammal* - This entity is also excluded from *basic level* membership, since its primary descendant has a higher *basic level* score than it does, i.e., dolphin has a score of 5 while mammal has a score of 4.
- *Dolphin* - The dolphin entity is chosen as the *basic level* entity since its *basic level* score is higher than its ancestors (dolphin = 5 > mammal = 4), its score is higher than its descendants (dolphin = 5 > toothless dolphin = 3), and it has a primary subset (dolphin HAS SUBSET toothless dolphin).
- *Toothless Dolphin* - This entity is also excluded from *basic level* membership, since it is never processed by the Restrictive Print algorithm. It's not processed because once the "Dolphin" entity was chosen as a *basic level* entity, processing was returned to the calling procedure, therefore never reaching the "Toothless Dolphin."

16.4. Output

The *basic level* entity output produced by the *Restrictive Print* algorithm is just a vertical list of entities labeled by the header "*Basic Level* Entities." An example is illustrated below that displays *basic level* entities selected by the *Restrictive Print* algorithm, after receiving a semantic net containing information about whales and dolphins.

```
Basic Level Entities
    whale#0
    dolphin#0
    tooth#0
    foot#0
    head#0
```

Example 16.2 - Basic Level Entities Sample Output

Basic-Level objects selected by the *basic-level* machine method are displayed in the boxes below. There is one box of *basic-level* objects for each article processed by the machine.

```
Basic Level Entities
   bear#0
```

Figure 16.2 - Basic Level Entities Selected by the Machine Method (Bear Article)

```
Basic Level Entities
   whale#0
   dolphin#0
   tooth#0
   foot#0
   head#0
```

Figure 16.3 - Basic Level Entities Selected by the Machine Method (Whale Article)

```
Basic Level Entities
   leg#0
   eye#0
   spider#0
   web#0
```

Figure 16.4 - Basic Level Entities Selected by the Machine Method (Spider Article)

16.5. Student Basic Level Survey

The medium used to validate the machine *basic level* method was a survey given to a class of college students. In the survey, one question was asked to calibrate the student's entity *basic level* selections after they had read a children's book on a particular animal category. The below question was used, because it was assumed that the students would not be able to grasp the meaning of what a *basic level* entity was within the given time frame allotted to answer the survey. It was hoped that students would use *basic level* entities in their response.

X) Write down in your own words, what you think the article was about.

Example 16.3 - Basic Level Selection Question Format

To evaluate the machine *basic level* method, the student responses to the above question needs to be converted into numerical data. This numerical conversion consists of several steps. First, the machine method receives the same article as the students were given, processes the article, and outputs a list of selected *basic level* entities. Next, an array of indicators is created for each student. Each position in the array represents an entity selected by the machine *basic level* method. The student responses are then read through and their array entries are marked with either 1s or 0s. An array entry will receive a value of 1 if the student mentioned the entity represented by that particular array entry, or a value of 0 if the student didn't mention that entity. The final result of this numerical conversion procedure is an m by n matrix of 1s and 0s, where m represents the number of students who participated in the survey and n is the number of entities selected by the machine *basic level* method. An example of this procedure is iterated below.

Numerical Conversion of Student Survey Basic Level Selections (*Machine Method Evaluation*)

- Two students, A and B, read an article about spiders and answered the question, "*Write down in your own words, what you think the article was about.*"
- The machine *basic level* method was also given the same article about spiders and displayed the following result:

```
Basic Level Entities
    leg
    eye
    spider
    web
```

- Student A's response to the previous question is displayed below. The entities that both the machine method displayed and the student wrote down are highlighted in bold and underlined.

  The article was about different types of **spiders**. It told of physical traits, behavior, reproduction, mating, and enemies.

- Student B's response to the previous question is displayed below. The entities that both the machine method displayed and the student wrote down are highlighted in bold and underlined.

  This article gives an introduction of the **spider**.

The **spiders** are divided into different kinds.
The **web** and mating are the center of this article.

- Displayed below are three arrays that represent the numerical consolidation of the machine method and students' responses. The top row shows the entities that the machine method selected as *basic level* entities. The next two rows display the student selections that matched the machine.

	Basic Level Entity Indicators			
Machine Basic Level Selection	*leg*	*eye*	*spider*	*web*
Student A	**0**	**0**	**1**	**0**
Student B	**0**	**0**	**1**	**1**

Example 16.4 - Numeric Conversion of Basic Level Results (Machine Method)

This numerical conversion was also applied to each student response to compute a standard. This standard indicates how hard it is for a machine method to choose entities that the student's mentioned in their responses. This calculation is similar to the one calculated for the machine method except for one difference. The difference is that all the objects a student mentions are recorded in a list and the ones not mentioned in the article are eliminated from that list. This replaces the list of *basic level* entities the machine method produced. From then on, it is the same as the numerical conversion that was calculated to get the arrays for the machine method evaluation. An example of this student evaluation procedure is iterated below.

Numerical Conversion of Student Survey Basic Level Selections (*Student Evaluation*)

- Two students, A and B, read an article about spiders and answered the question, "*Write down in your own words, what you think the article was about.*"
- Student A's response to the previous question is displayed below. The subjects and objects are highlighted in bold and underlined.

  The **article** was about different types of **spiders**. It told of **physical traits**, **behavior**, **reproduction**, **mating**, and **enemies**.

- Student B's response to the previous question is displayed below. The subjects and objects are highlighted in bold and underlined.

  This **article** gives an introduction of the **spider**.
  The **spiders** are divided into different **kinds**.
  The **web** and **mating** are the center of this **article**.

- Student A will be the student evaluated in this example. The subjects and objects the student mentioned in his response are listed below, with the ones not contained in the article ~~crossed out~~.

  ~~article~~
  spiders
  ~~physical traits~~
  behavior
  ~~reproduction~~
  mating
  enemies

- Displayed below are two arrays that represent the numerical consolidation of the selected student and the class's response. The top row shows the entities that student A mentioned in his response that were contained in the article. The next array row displays the entities the class mentioned

(*student B*) in their survey response. The matrix to be used to evaluate the selected student's (*student A*) performance is highlighted in bold.

	Basic Level Entity Indicators			
Selected Student's (Student A)	*spiders*	*behavior*	*mating*	*enemies*
Class Response (Student B)	**1**	**0**	**1**	**0**

Example 16.5 - Numeric Conversion of Basic Level Results (Student Method)

16.6. Validation of Machine Output

In the absence of a standardized testing instrument, measuring expert performance is difficult. Classically, human experts measure whether some new person is also an expert by how consistent the new person is with the recognized experts. A new person can be thought of as an expert if his performance is consistent with expert performance to the degree that the experts' performances are consistent with each other.

The objective of the following test is to measure to what extent a machine method can select *basic-level* objects. One way of determining this measure is by comparing the machine's *dual-group-consistency score* to the *dual-group-consistency score* of the class. The comparison is used to determine if the computer's *basic-level* selections are similar to a class of students' *basic-level* selections.

A *dual-group-consistency score* was developed for the machine. This was formulated by first, comparing the machine's *basic-level* selections to each student's *basic-level* selections. This resulted in an array of similarity scores. The similarity scores were then aggregated into one *dual-group-consistency score* by averaging them together. The idea is to discover how consistent the machine's *basic-level* selections were with the selections of the student aggregate. The consistency score indicated how similar the machine was to the student aggregate in selecting *basic-level* objects. A high average indicates a high level similarity.

A *dual-group-consistency score* was also developed for the student aggregate. This was done by first, comparing each student's *basic-level* selections with every other student's selections, excluding duplicate pairings. This resulted in array of similarity scores. The student similarity scores were then aggregated into one *dual-group-consistency score* by averaging them together. The idea being to discover how generally consistent the students were in their *basic-level* selections amongst themselves.

The extent that a machine can select *basic-level* objects can be measured by comparing the machine's *dual-group-consistency score* with the class's *dual-group-consistency score*. The scores were considered the same if they differed by no more than a tolerance value. In this experiment, the tolerance value was the variance of the student's' *dual-group-consistency score.*

16.6.1. Similarity Score

The similarity score between two sets of *basic-level* selections was measured by calculating the Pearson (Phi) Correlation Coefficient of the two. The Pearson statistic is used for two reasons. First, it indicates if two sets of numbers (two variables) are similar to one another. Second, it is the appropriate correlation coefficient to use when the two variables being compared are two valued variables.

The Pearson coefficient indicates similarity and dissimilarity between two variables with the following indicators. A Pearson coefficient with a value close to one indicates a high level of similarity between two variables. A value near zero indicates little or no similarity between the two variables. A value

close to negative one indicates selections that go in opposite directions. The formula for the Pearson Correlation Coefficient (similarity score) is detailed below. The Pearson correlation was applied wherever a similarity score was needed.

, where
n is the number of basic-level objects that can be selected and
x / y can be either machine/student selections or student/student selections.
[Mosteller, Rourke, 1973]

16.6.2. Machine Basic-Level Selection Evaluation Sample

To show how the machine method and class *dual-group-consistency* scores were derived, an example is calculated below. This example details how the *dual-group-consistency* score can be used to determine to what extent a machine method can select *basic-level* objects.

- First, five students were given an article about bears to read.
- Second, each student answered the question, "*Write down in your own words, what you think the article was about.*"
- After the students had finished their survey, the syntactic objects they wrote down in their replies to the above question, were tabulated into a table.
- Next, the machine *basic-level* method was also given the same article to process. It then indicates which objects in the article were *basic-level* objects.
- In the table below, the machine and the student selections are displayed. A "1" indicates that the machine or student had selected the object. A "0" indicates that the machine or student did not select the object. The students are labeled s1 through s5.

	machine	**Student Selections**				
Basic-level Objects	**selections**	**s1**	**s2**	**s3**	**s4**	**s5**
bear	1	1	1	1	1	1
live	0	1	0	0	0	0
cub	0	1	0	0	0	0
fur	0	1	0	0	0	0
brown bear	0	0	0	1	1	0
food	0	0	0	0	0	0

Table 16.1 - Machine/Student Sample Basic-Level Selections

- Next, the machine is correlated with all five students, using the Pearson Correlation Coefficient, to compute its similarity score. The students are then correlated with each other excluding any duplicate pairings, using the Pearson Correlation Coefficient, to compute their similarity scores. The result of these correlations is displayed in the table below:

	Machine	**Students**				
Students	**m**	**s1**	**s2**	**s3**	**s4**	**s5**
s1	0.32					
s2	1.00	0.32				

s 3	0.63	-0.25	0.63			
s 4	0.63	-0.25	0.63	1.00		
s 5	1.00	0.32	1.00	0.63	0.63	

Table 16.2 - Sample Basic-Level Similarity Scores

- The machine *dual-group-consistency score* is computed by averaging the correlation coefficients in column "m" (second column). The class's *dual-group-consistency score* is computed by averaging the correlation coefficients in columns "s1" through "s5". The tolerance value for the class's *dual-group-consistency score* is calculated by taking the variance of columns "s1" through "s5". The result of these three operations is displayed in the table below.

	Machine	**Students**
Mean	0.72	0.47
Variance		0.19

Table 16.3 - Basic-Level Dual-Group-Consistency Scores Example

- The *dual-group-consistency score*s are then used to evaluate how well the machine could select *basic-level* objects. In this example, the machine method had exceeded the students score by more than the tolerance value, therefor it was able to select *basic-level* objects better than the average student in this example.

16.6.3. Dual-Group-Consistency Score Test

The machine *basic-level* results were validated against three student surveys using the machine and student *dual-group-consistency* scores. The *dual-group-consistency* scores were calculated for each of the three surveys and the results recorded in the table below. The machine is considered to have performed the same as the average student if the machine and the class *dual-group-consistency* scores differed by no more than the tolerance value. If the machine's score exceeded the class's score by more than the tolerance value, then the machine is considered to have selected *basic-level* objects better than the average student. If the machine's score was below the class's score by more than the tolerance level, then the machine is considered to have performed worse than the average student in selecting *basic-level* objects.

A description of the *dual-group-consistency* score table's columns is detailed here. The "Survey" column displays which survey the evaluation results originated from. The "Machine *Dual-Group-Consistency* Score" column displays the machine's *basic-level* method's consistency scores. The "Class *Dual-Group-Consistency* Score" column displays the class's consistency scores. The "Tolerance Value" column displays the tolerance value corresponding to each class *dual-group-consistency* score.

Survey	**Machine *Dual-Group-Consistency* Score**	**Class *Dual-Group-Consistency* Score**	**Tolerance Value**
Bears	0.63	0.39	0.19
Whales & Dolphins	0.83	0.63	0.09
Spiders	0.65	0.64	0.06

Table 16.4 - Dual-Group-Consistency Score Machine Basic-Level Evaluation

The criterion for this performance evaluation is the machine's *basic-level* selection consistency with the students compared to the *basic-level* selection consistency among the students themselves. The class consistency score reflects what the average student's *dual-group-consistency* score. If the *basic-level* method is said to have performed as well as the average student, this means the method was just as consistent as the class was in selecting *basic-level* objects.

The machine *basic-level* method performed as well as or better than the average student in selecting *basic-level* objects. The *basic-level* method had higher *dual-group-consistency* scores than the class did in the bear and whale surveys. The method's consistency score was equal to the class's consistency score in the spider survey. The *basic-level* method can select *basic-level* objects as well as or better than the average student in this experiment.

A complete statistical summary of the student surveys and the machine *basic-level* output can be found in appendix J.

16.6.4. Investigator's View of the Machine's Basic-Level Selections

Initial observations of the machine's *basic-level* output indicated that the machine method did very well in selecting *basic-level* objects. It appeared that the machine method only selected basic-level objects and no others. *Basic-Level* objects are semantic primitives like "bear", "whale", "dolphin", "spider", and "web". The machine method did not pick objects like "food", "peg-leg tooth", "horny plate", "insect", "thing", "fruit" or "vegetable", objects that are not considered *basic-level* objects by definition.

One thing that might be improved in the machine's *basic-level* selection is the number of selected *basic-level* objects. The machine method missed a number of *basic-level* objects in each of the articles it processed. In the bear article the method missed objects like "snow", "mountain", and "termite". In the whale article, the method missed "elephant", shrimp" and "porpoise". In the spider article the method overlooked "bee", "wasp", and "maggot". Two ways that might increase the number of *basic-level* objects is to increase the amount of natural language text processed or add more definition to the machine dictionary.

16.7. Conclusion

Basic-level entities are summarized completely and their importance stated clearly with the quotation:

> *"Categorizations which humans make of the concrete world are not arbitrary but highly determined. In taxonomies of concrete objects, there is one level of abstraction at which the most basic category cuts are made. Basic categories are those which carry the most information, possess the highest category cue validity, and are, thus, the most differentiated from one another. ... basic objects are the most inclusive categories whose members: (a) possess significant numbers of attributes in common, (b) have motor programs which are similar to one another, (c) have similar shapes, and (d) can be identified from averaged shapes of members of the class."*
> [Rosch, Mervis, 1976, p. 382]

For a machine to show it has conceptualization capabilities it must be able to select *basic-level* entities from information it receives with accuracy.

To show the extent a machine method could select *basic-level* objects was a complex process. First, a machine method had to be developed to select *basic-level* objects. Second, tools had to be found that could evaluate the results of the *basic-level* method. Next, students had to be surveyed so there was

data to gauge the accuracy of the machine method in selecting *basic-level* entities. Finally, the results of the evaluations were collected and analyzed to determine if a machine with the appropriate method could select *basic-level* entities accurately.

The machine method that selected *basic-level* entities required two algorithms, *Calculate Basic Score* and *Restrictive Print*. The *Calculate Basic Score* algorithm calculated each entity's *basic-level* score within the net, based upon the number of arcs emanating from each entity and their primary ancestors. Selecting the *basic-level* entities, the *Restrictive Print* algorithm looked at the semantic net's primary children and applied a set of rules. The rules took into consideration each entity's primary ancestors and descendants to determine which ones would be printed out. The two algorithms together decided quickly and decisively which entities were to be *basic-level* entities.

A *dual-group-consistency score* was developed to evaluate the machine's *basic-level* method. The score essentially computes the similarity between the machine and human *basic-level* selections. The score was computed for both the machine and the class of students. The machine *dual-group-consistency score* entailed correlating the machine *basic-level* selections with each student's selections and averaging the coefficients into one score. The class *dual-group-consistency score* involved correlating each student's *basic-level* selections with every other student's selections, excluding duplicated pairings, and averaging the coefficients into one score. A tolerance value was also calculated by taking the variance of the class's correlation coefficients. The extent that a machine can select *basic-level* objects was measured by comparing the machine's *dual-group-consistency score* with the class's *dual-group-consistency score*. The scores were considered the same if they differed by no more than the tolerance value.

Three surveys were taken from a college student population to evaluate the performance of the machine's *basic-level* method. In each survey the students were handed a packet that consisted of an article and a set of questions. The article received by the students was about specific entities that they were to read about and to use as a reference in answering a summary question. The summary question the students were to answer had the following format, "*Write down in your own words, what you think the article was about,*" and it was used to capture the *basic-level* objects they might mention in their response. It was hoped that the students would use *basic-level* objects in summarizing the articles they read. The numerical data derived from the student responses were then used to validate the results of the machine's *basic-level* method.

The extent that a machine method could select *basic-level* objects was computed by comparing its *dual-group-consistency score* with the *dual-group-consistency score* of a class of students. In this experiment the *basic-level* method performed as well as or better than the average student in all three surveys. The *basic-level* method had higher *dual-group-consistency* scores than the class did in the bear and whale surveys. The method's consistency score was equal to the class's consistency score in the spider survey. This means that the machine selected *basic-level* objects with the same or more similarity than the average student. A machine can select *basic-level* objects to the extent of being the same or better than the average student.

17. Conclusion

For a computer to understand a natural language, it has to be able to conceptualize. Conceptualization involves many cognitive processes such as the ability to recognize related things, create a taxonomy, and determine a syntactic object's relative importance. To do this, it is useful to identify: categories, *best examples* of an object within a category, and underlying semantic building blocks.

This thesis measured to what extent a computer could conceptualize natural language text using an unsupervised semantic net formation technique. The things that this thesis attempted to do were:

- Measure to what extent a machine could categorize objects from natural language text.
- Determine to what extent a machine could rank natural language objects by importance.
- Gauge to what extent a machine could rank objects from natural language text by *best example*.
- Calibrate to what extent a machine could select basic-level objects from natural language text.

Several steps were performed for the computer conceptualization assessment. First, a computer program was written that could perform several of the aforementioned cognitive processes upon natural language text. The program creates a semantic net to formulate and identify concepts. Next, the program produced several conceptual results for each natural language text article it processed. Finally, the computer's performance was evaluated by taking its conceptualization results and comparing them to human conceptualization results.

Measuring to what extent a machine can conceptualize natural language text was done with four different evaluations. These four evaluations correspond to the cognitive processes of categorizing, determining importance, ranking *best example*s and selecting *basic-level* objects.

Categorizing

The investigator made observations about the machine's categorization output. The machine's categorization method was proficient at categorizing objects, since it did not place any object into a category that it was not a member of. It also induced a few categories that were not specifically mentioned in the natural language text input. For example, the categorization method created the "toothed dolphin" and "toothless dolphin" categories. On the other hand this can lead to some very generic categorizations.

In the experiments performed for this thesis there were a few categorizations that were too general. A few examples of this generic categorization were the creation of the "mother bear", "small whale", and "male spider" categories. Another problem recognized in the machine categorization output was the misplacement of the generic categories in the taxonomy. For example, "mother bear" was placed at the same level in the taxonomy as the "sloth bear" and "black bear". Another example was placing the "small whale" category at the same level as the "sperm whale" and the "humpback whale". Subsets of species should not be placed at the same level as the sub-species when creating a taxonomy.

Correcting the categorization problems experienced in this thesis could be handled in a couple of different ways. One way would be to provide the machine with more natural language text dealing with the subject to be categorized. A second way would be to define the dictionary in more detail, even though this goes against minimal dictionary data definition that was desired for this thesis. Another way would be to create additional relationships out of some of the sub categories and apply them to the sub category's taxonomy level. For example, take "mother bear" and place it underneath the sub species "brown bear" and "black bear", creating the "mother brown bear" and "mother black bear" sub-categories.

The verification of the extent that a machine could categorize a set of objects was computed by comparing its *dual-group-consistency score* with the *dual-group-consistency score* of a class of students. In this experiment the categorization method performed better than the average student in all three surveys. This means that the machine categorized a set of objects with more similarity than the average student when both their category selections were compared with the class's selections. A machine can categorize a set of related entities to the extent of being better than the average student.

Determining Importance

A machine provided with an importance ranking method could rank objects by importance very well. Looking at the machine's importance rankings, it seemed that the rankings made sense. For example, whales that were mentioned and described the most were located at the top of the rankings. Another observation about the machine's importance rankings is that it never gave the same rank to more than one fourth of the objects being ranked. This meant the machine method was able to differentiate between objects at a reasonable level.

One thing that could be improved in the machine importance ranking is the differential in rankings between animal subsets and animal sub species. At times, the importance method ranked species subsets along with the sub species. For example, the machine method ranked the "mother bear" higher than the "kodiak bear" and the same as "grizzly bear". There should have been some ranking difference between the specie subsets and the sub species.

In further importance ranking work, this problem of specie subsets and sub species importance ranking could be solved in a couple of ways. One way would be to create a different taxonomy structure that would exclude the specie subsets from the importance rankings. This is also another fix to the categorization problems that were experienced previously. Another solution would be to provide more information in either the form of natural language text or a more defined dictionary.

The measurement of the extent that a machine could rank a set of objects by importance was a twofold process. First, the best performing machine method had to be selected. This was done by using the *rank-individual-consistency* score. The consistency score picked the *All Ancestors* method as the best. Second, the *All Ancestors* method's ranking performance was evaluated in terms of the average student. In this experiment using the *rank-group-consistency* score as a gauge the following was indicated. The *All Ancestors* method could rank a set of objects by importance as well as or better than the average student. The average student being defined as a student who participated in the survey and was compared with the machine method and his fellow students in ranking objects by importance.

Best Example Ranking

Output of the machine's *best example* ranking method looked fairly well. Observing the machine's *best example* rankings, it seemed the rankings made sense after processing the natural language text, with one exception. For example, "brown bear", "polar bear" and "black bear" were the bears mentioned most often and first in describing bears and were selected as best examples. This was the same for the processed spider article with the "wolf" and "garden" spiders getting high best example rankings. The exception to the good rankings happened in the processing of the whale article with the "toothed whale" receiving the highest *best example* ranking. On the other hand, it was very positive to see the "blue whale" and the "humpback whale" come in second in the rankings. One other good observation about the machine's *best example* rankings was that it always gave out at least five different rankings.

There were two things that could make the *best example* ranking method better. The first thing that could be improved is the differential in rankings between animal subsets and animal sub species. At times, the *best example* method ranked species subsets along with and higher than the sub species. For example, the machine method ranked the "toothed whale" higher than any other whale when the whale

article was processed. When humans think of best examples they usually think of sub species, not a subset of species. This problem is related to the one experienced by the machine importance ranking method and the possible solutions are the same. Possible solutions to this sub specie and specie subset ranking problem would be to treat the sub species and specie subsets differently and process a higher volume of natural language text.

The second thing that would be nice to see in the *best example* rankings is a higher disparity in the rankings. In other words, the *best example* method should use more ranking numbers when ranking objects. This is especially true in the bear rankings where the rank of "10" was given out six times and in the spider rankings and the rank of "4" was given out eight times. A possible solution to this rank disparity problem is to process a higher volume of natural language text on the ranked subject matter. This would provide more information to differentiate between the different objects.

The measurement of the extent a machine can rank a set of objects by *best example* was a twofold process. First, the best performing machine method had to be selected. This was done using the *rank-individual-consistency* score and it picked the *Combination Method*. Second, the *Combination Method's* ranking performance was evaluated in terms of the average student. In this experiment using the *rank-group-consistency* score as a gauge the following was indicated. The *Combination Method* could rank a set of objects by *best example* as well as the average student. The average student being defined as a student who participated in the survey and was compared with machine method and his fellow students in ranking objects. If any of the other machine methods would have been selected as the best method to rank a set of objects by *best example*, then the extent that a machine could rank a set of objects by *best example* would have been inconclusive.

Basic-Level Selections

Initial observations of the machine's *basic-level* method output indicated that the machine method did very well in selecting *basic-level* objects. It appeared that the machine method selected only basic-level objects and no others. *Basic-Level* objects are semantic primitives like "bear", "whale", "dolphin", "spider", and "web". These objects were some of the ones selected by the machine *basic-level* method. The method did not pick objects like "food", "peg-leg tooth", "horny plate", "insect", "thing", "fruit" or "vegetable", objects that are not basic-level objects by definition.

One thing that could be improved in the machine *basic-level* selection method is the number of selected basic-level objects. The machine method missed a number of *basic-level* objects in each natural language text that it processed. In the bear article the method missed objects like "snow", "mountain", and "termite". In the whale article, the method missed "elephant", shrimp" and "porpoise". In the spider article the method overlooked "bee", "wasp", and "maggot". Two ways that might decrease the number of missed *basic-level* objects is to increase the amount of natural language text processed or add more definition to the machine dictionary.

The verification that a machine could select *basic-level* objects was computed by comparing its *dual-group-consistency score* with the *dual-group-consistency score* of a class of students. In this experiment the *basic-level* method performed as well as or better than the average student in all three surveys. The *basic-level* method had higher *dual-group-consistency* scores than the class did in the bear and whale surveys. The method's consistency score was equal to the class's consistency score in the spider survey. This means that the machine selected *basic-level* objects with the same or more similarity than the average student. A machine can select *basic-level* objects to the extent of being the same or better than the average student.

Summary

This thesis experimented with machine conceptualization of natural language text by forming a semantic net. It did this by breaking down conceptualization into identifiable cognitive processes.

Some of these cognitive processes were then emulated by different machine methods and tested for validation. Four cognitive processes associated with the conceptualization that were tested are categorization, importance determination, *best example* ranking, and *basic-level* identification. The summed up results of the conceptualization tests are listed below.

The investigator observed that:

- A machine with a categorization method can categorize objects with accuracy. On the other hand it created some generic categories and partitions at the wrong level in the taxonomy. This would later affect some of the other cognitive processes in a negative way.
- A machine using the *All Ancestors* importance method can determine object importance very well. It used the full range of rankings with disparity. It did have a problem in differentiating between animal sub species and specie subsets.
- A machine using the *Combination Method* can rank a set of objects by best example fairly well. However, if it is not provided with enough information it fails to differentiate between levels of *best exampleness*.
- A machine using the *basic-level* method is very accurate at selecting *basic-level* objects. Its one downfall is its failure to select all the *basic-level* objects within the natural language text.

The results of the verification experiment indicated that:

- A machine with a categorization method can categorize natural language text objects better than the average student.
- A machine with the *All Ancestors* method can rank a set of natural language text objects by importance as well as or better than the average student.
- A machine with the *Combination Method* can rank a set of natural language text objects by *best example* as well as the average student.
- A machine with a *basic-level* selection method can select *basic-level* natural language text objects as well as or better than the average student.

Viewing the cognitive tests as a whole, it can be said that machine with the appropriate methods can conceptualize natural language text as well as or better than the average student.

References

[Anderson & Matessa, 1991]
John R. Anderson & Michael Matessa. "An Incremental Bayesian Algorithm for Categorization", in Fisher, Pazzani, and Langley (Eds.) *Concept Formation: Knowledge and Experience In Unsupervised Learning,* Morgan Kaufmann Publishers, Inc. San Mateo, CA., 1991.

[Basili & Pazienza, 1993]
Roberto Basili & Maria Teresa Pazienza, "Relating diagrams to logic", in Mineau's (Ed.), *Conceptual Graphs for Knowledge Representation*, Springer-Verlag: Berlin, Germany, 1993.

[Berg, 1993]
Harmen van den Berg, "Modal Logics for Conceptual Graphs", in Mineau's (Ed.), *Conceptual Graphs for Knowledge Representation*, Springer-Verlag: Berlin, Germany, 1993.

[Berlin, Breedlove, Raven, 1974]
Brent Berlin, Dennis E. Breedlove, and Peter H. Raven. "Principles of Tzeltal Plant Classification", New York: Academic, 1974.

[Biewbow & Chaty, 1993]
Brigitte Biewbow & Guy Chaty, "A Comparison between Conceptual Graphs and KL-ONE", in Mineau's (Ed.), *Conceptual Graphs for Knowledge Representation*, Springer-Verlag: Berlin, Germany, 1993.

[Brachman, 1979]
Ronald J. Brachman. "On The Epistemological Status Of Semantic Networks" in Findler's (Ed.), *Associative Networks*, Academic Press Inc. New York, NY. 1979.

[Brachman, 1991]
Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, Lori Alperin Resnick and Alexander Borgida. "Semantic Nets Are in the Eye of the Beholder", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, John F. Sowa (Ed.), Morgan Kaufmann Publishers, Inc. San Mateo, CA, 1991.

[Bruner, Goodnow, Austin, 1956]
Jerome S. Bruner, Jacqueline J. Goodnow, & George A. Austin. *A Study of Thinking*, Wiley & Sons, New York, 1956.

[Carey, 1985]
Susan Carey. *Conceptual change in childhood*, Cambridge, MA: MIT Press, 1985.

[Cheng & Fu, 1985]
Yizong Cheng & King-Sun Fu. "Conceptual Clustering in Knowledge Organization", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, Vol. PAMI-7, No. 5, 1985, pp. 592-598.

[Clark, Clark, 1977]
Herbert H. Clark & Eve V. Clark, *Psychology and Language*, Harcourt Brace Jovanovich, New York, 1977.

[Crawford & Kuipers, 1991]

J. M. Crawford & Benjamin Kuipers, "ALL: Formalizing Access Limited Reasoning", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, John F. Sowa (Ed.), Morgan Kaufmann Publishers, Inc. San Mateo, CA, 1991.

[Creasy & Ellis, 1993]
Peter Creasy and Gerard Ellis, "A Conceptual Graphs Approach to Conceptual Schema Integration", in Mineau's (Ed.), *Conceptual Graphs for Knowledge Representation*, Springer-Verlag: Berlin, Germany, 1993.

[Cruse, 1977]
D. A. Cruse. "The Pragmatics of Lexical Specificity", *Journal of Linguistics* vol. 13, 1977, pp. 153-164.

[Dixon, 1982]
R. M. W. Dixon. *Where Have all the Adjectives Gone?*, Berlin: Walter de Gruyter, 1982.

[Ellis, 1993]
Gerard Ellis, "Efficient Retrieval from Hierarchies of Objects Using Lattice Operations", in Mineau's (Ed.), *Conceptual Graphs for Knowledge Representation*, Springer-Verlag: Berlin, Germany, 1993.

[Everitt, 1980]
Brian S. Everitt. *Cluster analysis*, London: Heinemann, 1980.

[Everitt, 1993]
Brian S. Everitt. *Cluster analysis*, Halsted Press: New York, 1993.

[Findler, 1979]
Nicholas V. Findler. *Associative Networks*, Academic Press Inc. New York, NY. 1979.

[Fisher & Pazzani, 1991]
Douglas H. Fisher, Jr., Michael J. Pazzani. "Computational Models of Concept Learning", in Fisher, Pazzani, and Langley (Eds.) *Concept Formation: Knowledge and Experience In Unsupervised Learning*, Morgan Kaufmann Publishers, Inc. San Mateo, CA., 1991.

[Fisher, Pazzani & Langley, 1991]
Douglas H. Fisher, Jr., Michael J. Pazzani, and Pat Langley (Eds.). *Concept Formation: Knowledge and Experience In Unsupervised Learning*, Morgan Kaufmann Publishers, Inc. San Mateo, CA., 1991.

[Foo, Garner & Rao, 1992]
Norman Foo, Brian J. Garner, Anand Rao, and Eric Tsui. "Semantic Distance in Conceptual Graphs, in T. Nagle, J Nagle and L. Gerholz, *Conceptual Structures : Current Research and Practice*, West Sussex : Ellis Horwood Limited, 1992. pp. 149 - 154.

[Gelman, 1988]
S. A. Gelman. "The development of induction within natural kind and artifact categories", *Cognitive Psychology*, 20, 1988, pp. 65-95.

[Gould, 1983]
Stephen Jay Gould. *Hen's Teeth and Horse's Toes*, found in Mazlack, New York: Norton, 1983.

[Harris, 1985]
Mary Dee Harris. *Introduction to Natural Language Processing*, Reston, Virginia - Reston Publishing Company, 1985.

[Hendrix, 1978]
Hendrix. "Partitioned Semantic Networks", in Donald Walker's, *Understanding Spoken Language*, Elsevier Science Publishing co., Inc., 1978, p. 126.

[Hendrix, 1979]
Gary G. Hendrix. "Encoding Knowledge in Partitioned Networks" in Findler's (Ed.), *Associative Networks*, Academic Press Inc. New York, NY. 1979.

[Iba & Gennari, 1991]
Wayne Iba & John H. Gennari. "Learning to Recognize Movements", in Fisher, Pazzani, and Langley (Eds.) *Concept Formation: Knowledge and Experience In Unsupervised Learning*, Morgan Kaufmann Publishers, Inc. San Mateo, CA., 1991.

[Kaindl, 1990]
H. Kaindl. "Tree Searching Algorithms", In Marsland and Schaeffer (Eds.) edition *Computers, Chess, and Cognition*, Springer-Verlag, New York, 1990.

[Kuipers, 1975]
Benjamin J. Kuipers. "A Frame for Frames". In D. G. Bobrow and A. M. Collins, eds., *Representation and Understanding: Studies in Cognitive Science*, New York: Academic Press, 1975.

[Kuno & Oettinger, 1962]
S. Kuno and A. G. Oettinger, "Multiple-Path Syntactic Analyzer," *Information Processing*, North-Holland, Amsterdam. 1962. pp. 306-312.

[Lackoff, 1987]
George Lackoff. *Women, Fire, and Dangerous Things*, Chicago: The University of Chicago Press, 1987.

[Levesque & Mylpoulos, 1979]
Hector Levesque and John Mylpoulos. "A Procedural Semantics for Semantic Networks," *Associative Networks*, Academic Press, 1979.

[Ling & Lee, 1992]
Tok-Wang Ling & Mong-Li Lee. "A Theory for Entity-Relationship View Updates", *Entity-Relationship Approach-ER*, Springer-Verlag: Berlin Germany, 1992.

[Macnamara, 1982]
John Macnamara. *Names for Things*, MIT Press, Cambridge, MA, 1982.

[Marek, Truszczynski, 1989]
Wiktor Marek and Miroslaw Truszczynski. "Relating auto epistemic and default logics", In Proceedings of the First International Conference on Principles of Knowledge Representations and Reasoning, Brachman, R. J., Levesque, H. J., and Reiter, R. (eds.), Morgan Kaufmann Publishers, San Mateo, CA, 1989, pp. 276-288.

[Marsland, Schaeffer, 1990]
T. Anthony Marsland and Jonathan Schaeffer (Eds.). *Computers, Chess, and Cognition*, Springer-Verlag: New York, 1990.

[Martin & Billman, 1991]

Joel D. Martin & Dorrit Billman. "Representational Specificity and Concept Learning", in Fisher, Pazzani, and Langley (Eds.) *Concept Formation: Knowledge and Experience In Unsupervised Learning*, Morgan Kaufmann Publishers, Inc. San Mateo, CA., 1991.

[Mayr, 1984]
Ernst Mayr. "Biological Classification: Toward a Synthesis of Opposing Methodologies", In Sober, 1984, pp. 646-662.

[Mazlack]
Lawrence J. Mazlack, "Taxonomic Ambiguities In Category Variations Needed To Support Machine Conceptualization".

[McDermott, 1978]
J. McDermott, A. Newell and J. Moore. "The Efficiency of Certain Production System Implementations". In D. A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems*, New York: Academic Press, 1978.

[McDonald, Hayes-Roth, 1978]
David McDonald and Frederick Hayes-Roth. "Inferential Searches of Knowledge Networks as an Approach to Extensible Language-Understanding Systems", In D. A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems*, New York: Academic Press, 1978.

[Merleau-Ponty, 1973]
Maurice Merleau-Ponty. "Consciousness and the Acquisition of Language", Northwestern University Press, Evanston, 1973.

[Mervis, 1987]
Carolyn Mervis. "Child-basic Object Categories and Early Lexical Development". In U. Neisser, ed., *Concepts and Conceptual Development: The Ecological and Intellectual Bases of Categorization*, New York: Cambridge University Press, 1987.

[Michalski, 1992]
R. S. Michalski. "Concept Learning," in Stuart Shapiro, *Encyclopedia of Artificial Intelligence*, New York: Wiley, 1992. pp. 249-259.

[Minsky, 1975]
Marvin Minsky. "A Framework for Representing Knowledge," in Patrick Henry Winston, *Psychology of Computer Vision*, McGraw Hill Book Company, 1975.

[Mosteller, Rourke, 1973]
Frederick Mosteller and Robert E. K. Rourke. Sturdy Statistics, Addison-Wesley Publishing Company, Reading, Massachusetts, 1973. pp. 290-291.

[Mugnier & Chein, 1993]
M. L. Mugnier & M. Chein, "Characterization and Algorithmic Recognition of Canonical Conceptual Graphs", in Mineau's (Ed.), *Conceptual Graphs for Knowledge Representation*, Springer-Verlag: Berlin, Germany, 1993.

[Newell, 1990]
A. Newell. *Unified theories of cognition*, Cambridge, MA: Harvard University Press, 1990.

[Norman, Rumelhart, 1975]
Donald A. Norman and David E. Rumelhart. Explorations in Cognition, W. H Freeman and Company: San Francisco, 1975.

[Odell, 1981]
Jack S. Odell. "Are Natural Language Interfaces Possible?", in Sowa's 1984 "Conceptual Structures: information processing in mind and machine", IBM Systems Research Institute, New York, 1981.

[Pirsig, 1974]
Robert M. Pirsig, *Zen and the Art of Motorcycle Maintenance*, Bantam Books: New York, 1974.

[Quinlan, 1986]
J. R. Quinlan. "Induction of decision trees", *Machine Learning*, 1, 1986, pp. 81-106.

[Ragavan, 1993]
Harish Ragavan, Larry Rendell, Michael Shaw, and Antoinette Tessmer. "Complex Concept Acquisition through direct search and Feature Caching," *Proc. Thirteenth Intl. Joint Conf. on AI.*, 1993. pp. 946 - 951.

[Rajasekar, Lobo, Minker, 1989]
Arcot Rajasekar, Jorge Lobo, and Jack Minker. "Skeptical Reasoning and Disjunctive Programs", In Proceedings of the First International Conference on Principles of Knowledge Representations and Reasoning, Brachman, R. J., Levesque, H. J., and Reiter, R. (eds.), Morgan Kaufmann Publishers, San Mateo, CA, 1989, pp. 157-169.

[Rauh & Stickel, 1992]
Otto Rauh & Eberhard Stickel. "Entity Tree Clustering - A method for simplifying ER Designs", *Entity-Relationship Approach-ER*, Springer-Verlag: Berlin Germany, 1992.

[Reich & Fenves, 1991]
Yoram Reich & Steven J. Fenves. "The Formation and Use of Abstract Concepts in Design", in Fisher, Pazzani, and Langley (Eds.) *Concept Formation: Knowledge and Experience In Unsupervised Learning*, Morgan Kaufmann Publishers, Inc. San Mateo, CA., 1991.

[Richman, 1991]
Howard B. Richman. "Discrimination Net Models of Concept Formation", in Fisher, Pazzani, and Langley (Eds.) *Concept Formation: Knowledge and Experience In Unsupervised Learning*, Morgan Kaufmann Publishers, Inc. San Mateo, CA., 1991.

[Rosch, Mervis, 1975]
E. Rosch & C. Mervis. "Family resemblances: Studies in the internal structure of categories", *Cognitive Psychology*, 7, 1975, pp. 573-605.

[Rosch, 1975]
Eleanor Rosch. "Cognitive Reference Points", *Cognitive Psychology*, 1975, vol. 7, pp. 532-547.

[Rosch, 1976]
E. Rosch, C. Mervis, W. Gray, D. Johnson, & P. Boyes-Braem. "Basic objects in natural categories, *Cognitive Psychology*, 18, 1976, pp. 382-439.

[Rumelhart, 1972]
D. H. Rumelhart, P. H. Lindsay, and D. A. Norman, "A process model of Long-Term memory", In *Organization of Memory*, E. Tulving and W. Donaldson (eds.) Academic Press: New York, 1972.

[Schank, 1973]

R. C. Schank. "Identification of conceptualizations underlying natural language", In *Computer Models of Thought and Language*, Schank R. C. & Colby K. M. (eds.), W. H. Freeman and Company, San Francisco, CA., 1973.

[Schank, 1975]
R. Schank. "Conceptual Information Processing," with contributions from N. Golman, C. Rieger, and C. Riesbeck, *Vol. 3 of Fundamental Studies in Computer Science*, North-Holland, Amsterdam, 1975.

[Schank, Abelson, 1977]
Roger C. Schank and Robert Abelson. *Scripts, Plans, Goals and Understanding*, Hillsdale, New Jersey - Lawrence Erlbaum Associates, Inc., 1977.

[Schank, Riesbeck, 1981]
Roger C. Schank & Christopher K. *Riesbeck. Inside Computer Understanding: Five Program Plus Miniatures*, Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1981.

[Schmucker, 1984]
Kurt J. Schmucker. *Fuzzy Sets, Natural Language Computations, and Risk Analysis*, Rockville, Maryland: Computer Science Press, 1984.

[Schubert, 1991]
Lenhart K. Schubert, "Semantic Nets Are in the Eye of the Beholder", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, John F. Sowa (Ed.), Morgan Kaufmann Publishers, Inc. San Mateo, CA, 1991.

[Scott & Markovitch, 1991]
Paul D. Scott & Shaul Markovitch. "Representation Generation in an Exploratory Learning System", in Fisher, Pazzani, and Langley (Eds.) *Concept Formation: Knowledge and Experience In Unsupervised Learning*, Morgan Kaufmann Publishers, Inc. San Mateo, CA., 1991.

[Searle, 1978]
John Searle. "The Philosophy of Language", in Bryan Magee eds., *Men of Ideas*, Oxford University Press, New York, 1978, pp. 153-172.

[Selman & Levesque, 1991]
Bart Selman & Hector J. Levesque, "The Tractability Of Path-Based Inheritance", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, John F. Sowa (Ed.), Morgan Kaufmann Publishers, Inc. San Mateo, CA, 1991.

[Shapiro, 1979]
Stuart C. Shapiro. "The SNePS Semantic Network Processing System" in Findler's (Ed.), *Associative Networks*, Academic Press Inc. New York, NY. 1979.

[Shapiro, 1991]
Stuart C. Shapiro, "Cables, Paths, And 'Subconscious' Reasoning' In Propositional Semantic Networks", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, John F. Sowa (Ed.), Morgan Kaufmann Publishers, Inc. San Mateo, CA, 1991.

[Shapiro, Rapaport, 1992]
Stuart C. Shapiro and William J. Rapaport, "The SNePS Family", *Computer Mathematical Applications*, Vol. 23, 1992. pp. 243-275

[Shapiro, 1992]

Stuart C. Shapiro (Ed.). *Encyclopedia Of Artificial Intelligence Second Addition - Volume 1*, John Wiley & Sons, Inc., New York, 1992.

[Shastri, 1991]
Lokendr Shastri, "Why Semantic Networks", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, John F. Sowa (Ed.), Morgan Kaufmann Publishers, Inc. San Mateo, CA, 1991.

[Shuldberg, 1993]
H. Kelly Shuldberg, Melissa Macpherson, Pete Humphrey, and Jami Corley. "Distilling Information from Text: The EDS Templatefiller System," *Journal of the American Society for Information Science*, October, 1993. pp. 493-507.

[Simmons, 1973]
Robert F. Simmons. "Semantic Networks: Their Computation and Use for Understanding English Sentences", *Computer Models of Thought and Language*, ed. Roger Schank and Kenneth M. Colby, San Francisco: Freeman and Company, 1973.

[Smith, Medin 1981]
Edward E. Smith & Douglas L. Medin. "Categories and Concepts", Harvard University Press, Cambridge, MA, 1981.

[Sober, 1984]
Elliott Sober (Ed.). Conceptual Issues in Evolutionary Biology, The MIT Press, Cambridge, Massachusetts, 1984.

[Song, Johannesson & Bubenko, 1992]
William W. Song, Paul Johannesson, & Janis A. Bubenko Jr.. "Semantic Similarity Relations in Schema Integration", *Entity-Relationship Approach-ER*, Springer-Verlag: Berlin Germany, 1992.

[Sowa, 1984]
John F. Sowa. "Conceptual structures : information processing in mind and machine", Reading, Mass : Addison-Wesley, 1984.

[Sowa, 1991]
John F. Sowa (Ed.). *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann Publishers, Inc. San Mateo, California, 1991.

[Sowa, 1993]
John F. Sowa, "Relating diagrams to logic", in Mineau's (Ed.), *Conceptual Graphs for Knowledge Representation*, Springer-Verlag: Berlin, Germany, 1993.

[Stein, 1991]
Lynn Andrea Stein, "Extensions Of Possible Worlds", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, John F. Sowa (Ed.), Morgan Kaufmann Publishers, Inc. San Mateo, CA, 1991.

[Thalheim, 1992]
Bernhard Thalheim. "Fundamentals of Cardinality Constraints", *Entity-Relationship Approach-ER*, Springer-Verlag: Berlin Germany, 1992.

[Thorton, 1992]
C. J. Thornton. *Techniques in Computational Learning An Introduction*, Chapman & Hall Computing: London, 1992.

[Touretzky, 1986]
David S. Touretzky. *The Mathematics of Inheritance Systems*, Morgan Kaufmann Publishers, Inc., Los Altos, California, 1986.

[Utgoff, 1989]
P. E. Utgoff, "Incremental induction of decision trees", *Machine Learning*, 4, 1989, pp. 161-186.

[Willems, 1993]
M. Willems, "A Conceptual Semantics Ontology for Conceptual Graphs", in Mineau's (Ed.), *Conceptual Graphs for Knowledge Representation*, Springer-Verlag: Berlin, Germany, 1993.

[Winston, 1992]
Patrick Henry Winston. *Artificial Intelligence*, Addison-Wesley: Reading, Mass., 1992.

[Wittgenstein, 1953]
Ludwig Wittgenstein, *Philosophical Investigations*, in Sowa's 1984 "Conceptual Structures: information processing in mind and machine", Basil Blackwell, Oxford, 1953.

[Woods, 1991]
W. A. Woods, "Understanding Subsumption And Taxonomy: A Framework for Progress", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, John F. Sowa (Ed.), Morgan Kaufmann Publishers, Inc. San Mateo, CA, 1991.

[Zadronzy, 1991]
Wlodek Zadrozny, "Logical Dimensions of Some Graph Formalisms", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, John F. Sowa (Ed.), Morgan Kaufmann Publishers, Inc. San Mateo, CA, 1991.

[Zhou & Baumann, 1992]
Jian Zhou and Peter Baumann. "Evaluation of Complex Cardinality Constraints", *Entity-Relationship Approach-ER*, Springer-Verlag: Berlin Germany, 1992.

Appendix A: Attribute Table

[action,no]
[action,yes]
[caste,integer]
[caste,real]
[caste,alphanumeric]
[caste,alphabetic]
[caste,fraction]
[collection,mass]
[collection,single]
[collection,both]
[collection,collective]
[collection,part]
[count,single]
[count,plural]
[count,both]
[definition,yes]
[definition,no]
[description,abstract]
[description,concrete]
[description,animate]
[description,inanimate]
[description,location]
[description,edible]
[description,negative]
[gender,male]
[gender,female]
[gender,dual]
[gender,neutral]
[gender,all]
[instance,no]
[instance,yes]
[name,no]
[name,yes]
[neg adj,toothless#0]
[negative,no]
[negative,yes]
[number type,position]
[number type,number]
[number type,fraction]
[number type,multiplier]
[number type,quantity]
[person,first]
[person,second]
[person,third]
[pos adj,toothed#0]
[pos adj,plated#0]
[pronoun type,personal]
[pronoun type,reflexive]
[pronoun type,possessive]
[pronoun type,relative]
[pronoun type,interrogative]
[pronoun type,demonstrative]
[pronoun type,universal]
[pronoun type,assertive]
[pronoun type,negative]

```
[specific,no]
[specific,yes]
[tense,past]
[tense,present]
[tense,future]
```

Appendix B: Student Survey

Directions

Introduction : This questionnaire surveys how you recognize:

- groups of similar things. These are called categories.
- the *best exampl*e of something in a category.
- the *most important* thing found in a category.

Definitions

- ***Best Example*** - The *Best Example* is the object that gives the best general description for the rest of the objects in the category. It also can be the object used as a stereotypical example for the rest of the objects in the category.

  For example:
  Given a group of houses:
  cape cod house, White House, single story ranch, Fred's house,
  school house, and a two story red brick house
  The *Best Example* of a house selected could be a single story ranch.

- ***Most Important*** - The *Most Important* object is the object that:
  - exemplifies the best traits
  - has the most known about it
  - does the most

  For example:
  Given a group of houses:
  cape cod house, White House, single story ranch, Fred's house,
  school house, and a two story red brick house
  The *Most Important* house selected could be the White House.

Directions

- Read the attached article.
- Answer questions 1 through 8.
- After answering questions 1 through 8, hand the questionnaire (except for the first and second pages) back to the proctor and receive question 9.
- Answer question 9.
- It is important to follow the instructions exactly in order to maintain the survey's validity.

Articles

Bears

Brown bears are found in America, Europe, and Asia. Not all Brown bears are brown. Some are cream, some are light brown, and some are almost black like the Blue bear. The Kodiak bear from Alaska is the biggest kind of Brown bear. The Grizzly bear from North America is probably the fiercest. Here are some kinds of Brown bear.

In cold countries, bears hibernate throughout the winter. They find a hole or cave, called a den, in which they sleep. The babies, called cubs, are born in the den in midwinter. The mother and cubs do not leave the den until springtime.

Black bears live in the woods and forest of North America. They are much smaller than Brown bears. Black bears are very mischievous and often steal food from campers and picnickers. They like eating sweet things.

Black bears are also found the mountains and forest of Asia. They are sometimes called Moon bears because they have white mark shaped like a new moon on their chests. Although they feed mainly on fruit and vegetables, they will kill and eat any animal they can catch.

Bears are very fond of honey. This Black bear is robbing a wild bee's nest. Although he gets stung he does not seem to mind to much.

The Spectacled bear from South America eats only plants. The mother Sloth bear from India enjoys tiny insects called termites which she digs from their nest with her powerful claws. The mother bear often carries her cubs around on her back.

The Sun bear who lives in South East Asia is the smallest bear of all. He eats fruit. He is a very good climber.

Polar bears live in the Arctic. For much of the year the land and sea are covered with ice and snow. Polar bears enjoy the fruits and berries that grow in the short Arctic summer, but for most of the year they are flesh eaters.

Polar bears hunt seals. Their white fur makes them difficult to see against the ice and snow. Polar bears are also excellent swimmers. They travel far out to sea, often resting on floating ice.

This mother bear has caught a young seal for food for her cubs. She protects the cubs from the male Polar bear who will eat them if he can catch them.

Bears do not live in families. These cubs will stay with their mother until they are old enough to hunt for food for themselves.

Whales & Dolphins

Whales and dolphins have fins and live in the sea, but they are not fish. They are mammals. They must rise to the surface of the water to breathe air.

Whales, dolphins, and porpoises are members of the same animal family. Some of the family have teeth. Others do not. They hunt their food underwater.

Blue whales can stay underwater for more than half an hour. They may be as heavy as twenty elephants. Blue whales can swim almost twenty miles per hour.

Dolphins send out sounds to find food. The sounds bounce back from whatever they hit. Humans cannot hear them.

Toothless whales have horny plates call baleen. The plates have bristles that hold in shrimp and other food.

Humpback whales send out special songs that can last twenty minutes. Humpback whales can leap out of the water. The leaps are called breaching.

Whales have tough, spongy skin. The fat under the skin is called blubber. It keeps whales warm in cold water.

Whale calves are born underwater, tail first. Their mothers push them to the surface of the water to breathe.

Most whales travel the same route through the seas every year. Whales and dolphins like to stay with their own kind. Some even travel in herds of up to one thousand.

Whales can be friendly toward humans. Some allow people to pet them. Divers swim safely near most whales and calves. Dolphins are also friendly. There are many stories about dolphins rescuing humans. Dolphins are easily trained.

Whales sometimes strand themselves on beaches. They die out of the water. Entire herd of whales have stranded themselves. No one knows exactly why.

The largest of all toothed whales is the sperm whale. Sperm whales can grow to 65 feet long and weight almost 40 tons. They can dive to almost 10,000 feet.

Orca whales live all over the world. They eat sharks, small whales, and sea lions. They tear their victims with their teeth. Orcas are called killer whales because they helped early Spanish whalers to hunt other whales. They are very smart.

Dolphins have beaked heads and peg-leg teeth. Porpoises have rounder heads and spade-like teeth. In some countries, it is not legal to kill dolphins and whales.

Spiders

Have you noticed that spiders have eight legs? Usually they have eight eyes too. A spider is an "arachnid" and not an insect. Insects have only six legs.

There are 35,000 different kinds of spider. They have to hunt other creatures for their food, but very few of them will hurt you. This House Spider has trapped a fly in its cobweb.

A spider weaves a web from silk, which comes from its "spinnerets." They are like six little jets on the underside of its body. The silk is stronger than steel wire of the same thickness. Spiders' webs are almost invisible, but you can see them on a dewy or frosty morning. Threads of gossamer are left by spiders as they jump or fly off on the breeze.

Different sorts of spiders weave different webs. The Garden Spider always makes its web like this. First it attaches the corner threads to twigs or leaves. Then it makes three "spokes" of the wheel. It strengthens the center with a spiral frame. Finally it weaves the sticky main spiral, working inward from the outside. The spider sits in the middle. It can fell if anything touches the web.

Now the spider waits for an insect to fly into the web and be trapped in the sticky threads. The Hammock Spider can feel the vibrations when an insect drops down into the web. This spider has injected a poison with its fans. Now it binds the insect with sticky thread. Spiders eat by sucking out all the juices of their victims.

Male spiders are usually much smaller than the females. This Garden Spider is visiting a female in her web. He taps out a message to let her know who is calling. He doesn't want to be her lunch.

The female Malmignette often eats her mate. She is closely related to the American Black Widow. Both spiders can kill a human with a bite from their poisonous fangs.

All spiders hatch from eggs. A mother spider may lay as few as two eggs or as many as 2,000. She makes a cocoon from silk. This is a Golden Garden Spider's egg-sag.

Wolf spiders have no webs as homes, so they carry their egg-sacs around with them. When the spiderlings hatch, they ride around on their mothers' backs. This one has about forty babies. Wolf spiders don't build webs to trap insects. They chase after them. Wolf spiders can run very fast and they have sharp eyes. You often see them scuttling across the ground.

Did you know that a Tarantula looks likes this? Its bite is poisonous, but not deadly. Like all wolf spiders, the male attracts the female with a special dance.

The Trapdoor Spider comes from America. Its home is a burrow lined with silk. The spider lies in wait at the entrance. The Funnel-web of Australia is another sort of trapdoor spider. It weaves a funnel-shaped web. Australian children know not to touch this spider, as its bite could kill them.

Water Spiders make their homes under water. They carry bubbles of air around with them so they can breathe. Mostly they catch insects. This on is eating a tiny minnow. The Water Spider's web is like a diving bell anchored to the water weeds. The female fills it with bubbles of air. She lays her eggs in it and goes to sleep there in the winter.

The little Zebra Spider pounces on insects. It is a jumping spider. It trails a line of silk behind it to save it from falling. Watch for them on a tree stump or a garden wall.

Crab spiders scuttle sideways like crabs. Some of them are brightly colored, like the flowers they hide in. They can grab butterflies and bees that are hunting for nectar.

Spiders have many enemies. They are eaten by all sorts of animals, as well as other spiders. This wasp paralyzes the spider and lays an egg on it. The maggots feed on the live spider. Although a mother spider may lay hundreds of eggs, only one or two will grow up to have babies of their own. One huge pet Bird-eating spider lived to be twenty years old.

Questions

Bear Survey Questions

1) Rank the following objects according to what you learned from the article. Rank them in descending order. Give the object you are most knowledgeable about a "1" and the one you know least about a rank of "7". *All objects must be ranked.*

___ polar bear
___ brown bear
___ sun bear
___ kodiak bear
___ sloth bear
___ grizzly bear
___ black bear

2) Rank the following objects according to what is the ***Best Example*** of a bear from the previous article. Rank them in descending order. Give the *Best Example* object a "1" and the worst example a rank of "7". *All objects must be ranked.*

___ polar bear
___ brown bear
___ sun bear
___ kodiak bear
___ sloth bear
___ grizzly bear
___ black bear

3) Rank the following objects according to what is the ***Most Important*** bear from the previous article. Rank them in descending order. Give the *Most Important* object a "1" and the least important a rank of "7". *All objects must be ranked.*

___ polar bear
___ brown bear
___ sun bear
___ kodiak bear
___ sloth bear
___ grizzly bear
___ black bear

Use the knowledge you learned from reading the previous article to answer the following questions. *If you are **not** sure of the category, **don't** circle it.*

4) Circle the letter of the following category(s) that include polar bears.
   a. brown bears
   b. bears
   c. black bears
   d. flesh eaters
   e. biggest brown bears

5) Circle the letter of the following category(s) that include kodiak bears.
   a. brown bears
   b. bears
   c. black bears
   d. flesh eaters
   e. biggest brown bears

6) Circle the letter of the following category(s) that include moon bears.
   a. brown bears
   b. bears
   c. black bears
   d. flesh eaters
   e. biggest brown bears

7) Circle the letter of the following category(s) that include grizzly bears.
   a. brown bears
   b. bears
   c. black bears
   d. flesh eaters
   e. biggest brown bears

8) Circle the letter of the following category(s) that include sloth bears.
   a. brown bears
   b. bears
   c. black bears
   d. flesh eaters
   e. biggest brown bears

9) Write down, in your own words, what you think the article was about.

Whale & Dolphin Survey Questions

1) Rank the following objects according to what you learned from the article. Rank them in descending order. Give the object you are most knowledgeable about a "1" and the one you know least about a rank of "7". *All objects must be ranked.*

___ toothed whale
___ toothless whale
___ blue whale
___ humpback whale
___ orca whale
___ small whale
___ sperm whale

2) Rank the following objects according to what is the ***Best Example*** of a whale from the previous article. Rank them in descending order. Give the *Best Example* object a "1" and the worst example a rank of "7". *All objects must be ranked.*

___ toothed whale
___ toothless whale
___ blue whale
___ humpback whale
___ orca whale
___ small whale
___ sperm whale

3) Rank the following objects according to what is the ***Most Important*** whale from the previous article. Rank them in descending order. Give the *Most Important* object a "1" and the least important a rank of "7". *All objects must be ranked.*

___ toothed whale
___ toothless whale
___ blue whale
___ humpback whale
___ orca whale
___ small whale
___ sperm whale

Use the knowledge you learned from reading the article to answer the following questions. *If you are **not** sure of the category, **don't** circle it.*

4) Circle the letter(s) of the following category(s) that include blue whales.
   a. whales
   b. dolphins
   c. toothed whales
   d. toothless whales
   e. mammals

5) Circle the letter(s) of the following category(s) that include sperm whales.
   a. whales
   b. dolphins
   c. toothed whales
   d. toothless whales
   e. mammals

6) Circle the letter(s) of the following category(s) that include orca whales.
   a. whales
   b. dolphins
   c. toothed whales
   d. toothless whales
   e. mammals

7) Circle the letter(s) of the following category(s) that include humpback whales.
   a. whales
   b. dolphins
   c. toothed whales
   d. toothless whales
   e. mammals

8) Circle the letter(s) of the following category(s) that include toothed dolphins.
   a. whales
   b. dolphins
   c. toothed whales
   d. toothless whales
   e. mammals

9) Write down, in your own words, what you think the article was about.

Spider Survey Questions

1) Rank the following objects according to what you learned from the article. Rank them in descending order. Give the object you are most knowledgeable about a "1" and the one you know least about a rank of "7". *All objects must be ranked.*

___ house spider
___ garden spider
___ hammock spider
___ malmignette
___ black widow
___ wolf spider
___ crab spider

2) Rank the following objects according to what is the ***Best Example*** of a spider from the previous article. Rank them in descending order. Give the *Best Example* object a "1" and the worst example a rank of "7". *All objects must be ranked.*

___ house spider
___ garden spider
___ hammock spider
___ malmignette
___ black widow
___ wolf spider
___ crab spider

3) Rank the following objects according to what is the ***Most Important*** spider from the previous article. Rank them in descending order. Give the *Most Important* object a "1" and the least important a rank of "7". *All objects must be ranked.*

___ house spider
___ garden spider
___ hammock spider
___ malmignette
___ black widow
___ wolf spider
___ crab spider

Use the knowledge you learned from reading the article to answer the following questions. *If you are **not** sure of the category, **don't** circle it.* You may circle more than one letter per question.

4) Circle the letter(s) of the following category(s) that include black widows.
   a. spiders
   b. insects
   c. trapdoor spiders
   d. jumping spiders
   e. arachnids

5) Circle the letter(s) of the following category(s) that include wolf spiders.
   a. spiders
   b. insects
   c. trapdoor spiders
   d. jumping spiders
   e. arachnids

6) Circle the letter(s) of the following category(s) that include zebra spiders.
   a. spiders
   b. insects
   c. trapdoor spiders
   d. jumping spiders
   e. arachnids

7) Circle the letter(s) of the following category(s) that include funnel-webs.
   a. spiders
   b. insects
   c. trapdoor spiders
   d. jumping spiders
   e. arachnids

8) Circle the letter(s) of the following category(s) that include garden spiders.
   a. spiders
   b. insects
   c. trapdoor spiders
   d. jumping spiders
   e. arachnids

9) Write down, in your own words, what you think the article was about.

Appendix C: Phrase Parse Automata

Automaton State Descriptions

State Q0 - *Initial State*

- Create a new phrase node and add it to the current phrase section.

State Q1 - *Preposition*

- Add the preposition as a premodifier of the subject.
- Increment the current definition pointer by one and go to the next indicated state.

State Q2 - *Determiner*

- Add the determiner as a premodifier of the subject.
- Increment the current definition pointer by one and go to the next indicated state.

State Q3 - *Adjective*

- Add the adjective as a premodifier of the subject.
- Increment the current definition pointer by one and go to the next indicated state.

State Q4 - *Noun or Pronoun*

- Add the noun or pronoun to the subject of the current phrase node.
- Increment the current definition pointer by one and go to the next indicated state.

State Q5 - *Premodifier*

- Add the premodifier as a premodifier of the subject.

- Increment the current definition pointer by one and go to the next indicated state.

State Q6 - *Conjunction or Comma*

- Increment the current definition pointer by one and go to the next indicated state.

State Q7 - *Conjunction or Comma*

- If current word is a comma and the grammar of the previous word was a noun or adjective, and if the grammar of the following word is a noun and the grammar of the word after that is a verb: add the current subject as a premodifier to the subject and delete only the current subject.
- Increment the current definition pointer by one and go to the next indicated state.

State Q8 - *Negative*

- Add the negative as a premodifier of the subject.
- Increment the current definition pointer by one and go to the next indicated state.

State Q9 - *Preposition*

- Add the preposition as a post-modifier of the subject.
- Increment the current definition pointer by one and go to the next indicated state.

State Q10 - *Determiner*

- Add the determiner as a post-modifier of the subject.
- Increment the current definition pointer by one and go to the next indicated state.

State Q11 - *Noun*

- Add the noun as a post-modifier of the subject.

- Increment the current definition pointer by one and go to the next indicated state.

State Q12 - *Conjunction*

- Increment the current definition pointer by one and go to the next indicated state.

State Q13 - *Interjection*

- Increment the current definition pointer by one and go to the next indicated state.

State Q14 - *Pronoun representing the subject*

- Add the pronoun to the subject of the current phrase node.
- Increment the current definition pointer by one and go to the next indicated state.

State Q15 - *Noun*

- If the noun has a single count and the current subject has a plural count: add the noun as a premodifier of the subject.
- Otherwise, add the current subject as a premodifier to the subject, delete only the current subject and make the noun the subject of the current phrase node.
- Increment the current definition pointer by one and go to the next indicated state.

State Q16 - *Conjunction*

- Increment the current definition pointer by one and go to the next indicated state.

State Q17 - *Pronoun acting as the subject*

- Increment the current definition pointer by one and go to the next indicated state.

State Q18 - *Determiner*

- Add the determiner as a pre-modifier of the subject.
- Increment the current definition pointer by one and go to the next indicated state.

State Q19 - *Adverb*

- Add the adverb as a pre-modifier of the action.
- Increment the current definition pointer by one and go to the next indicated state.

State Q20 - *Auxiliary*

- Add the auxiliary as a pre-modifier of the action.
- Increment the current definition pointer by one and go to the next indicated state.

State Q21 - *Negative*

- Add the negative as a pre-modifier of the action.
- Increment the current definition pointer by one and go to the next indicated state.

State Q22 - *Verb*

- Add the verb as a to the action of the current phrase node.
- Increment the current definition pointer by one and go to the next indicated state.

State Q23 - *Adverb*

- Add the adverb as a post-modifier of the action.
- Increment the current definition pointer by one and go to the next indicated state.

State Q24 - *Negative*

- Add the negative as a post-modifier of the action.
- Increment the current definition pointer by one and go to the next indicated state.

State Q25 - *Noun or Pronoun - Object*

- If the current word is a noun and the following word is a noun, add the noun as pre-modifier of the object.
- Otherwise, add the noun or pronoun to the object of the current action.
- Increment the current definition pointer by one and go to the next indicated state.

State Q26 - *Adjective*

- If the next word is equal to "or" add the adjective as pre-modifier of the object.
- If the next word is a comma or a conjunction (not equal to "or"), add the adjective as an object of the current action.
- Increment the current definition pointer by one and go to the next indicated state.

State Q27 - *Determiner*

- Add the determiner as a pre-modifier of the object.
- Increment the current definition pointer by one and go to the next indicated state.

State Q28 - *Preposition*

- If the grammar of the previous word was a verb, add the preposition as a post-modifier to the current verb. Otherwise add the preposition as a pre-modifier of the object.
- Increment the current definition pointer by one and go to the next indicated state.

State Q29 - *Premodifier*

- Add the premodifier as a pre-modifier of the object.
- Increment the current definition pointer by one and go to the next indicated state.

State Q30 - *Comma*

- Increment the current definition pointer by one and go to the next indicated state.

State Q31 - *Conjunction*

- Increment the current definition pointer by one and go to the next indicated state.

State Q32 - *Preposition*

- Add the preposition as a post-modifier of the object.
- Increment the current definition pointer by one and go to the next indicated state.

State Q33 - *Adjective*

- Add the adjective as a post-modifier of the object.
- Increment the current definition pointer by one and go to the next indicated state.

State Q34 - *Noun or Pronoun*

- Add the noun or pronoun as a post-modifier of the object.
- Increment the current definition pointer by one and go to the next indicated state.

State Q35 - *Determiner*

- Add the determiner as a post-modifier of the object.
- Increment the current definition pointer by one and go to the next indicated state.

State Q36 - *Preposition*

- Add the preposition as a post-modifier of the object.
- Increment the current definition pointer by one and go to the next indicated state.

State Q37 - *Determiner*

- Add the determiner as a post-modifier of the object.
- Increment the current definition pointer by one and go to the next indicated state.

State Q38 - *Noun or Pronoun*

- Add the noun or pronoun as a post-modifier of the object.
- Increment the current definition pointer by one and go to the next indicated state.

State Q39 - *Adverb*

- Add the adverb as a post-modifier of the action.
- Increment the current definition pointer by one and go to the next indicated state.

State Q40 - *Premodifier*

- Add the premodifier as a post-modifier of the object.
- Increment the current definition pointer by one and go to the next indicated state.

State Q41 - *Conjunction*

- Add the conjunction as a post-modifier of the object.
- Increment the current definition pointer by one and go to the next indicated state.

State Q42 - *Adjective*

- Add the adjective as a post-modifier of the subject.
- Increment the current definition pointer by one and go to the next indicated state.

State Q43 - *Preposition*

- Add the preposition as a post-modifier of the action.
- Increment the current definition pointer by one and go to the next indicated state.

State Q44 - *Conjunction*

- If the grammar of the next word is a noun then create a new phrase node, increment the current definition pointer by one, and go to state Q0.
- Otherwise, increment the current definition pointer by one and go to the next indicated state.

State Q45 - *Verb or Auxiliary - Create a modifying phrase*

- Add a modifying phrase node to the current object.
- Set the current phrase equal to the phrase node created in the previous step.
- Decrement the current definition pointer by one.
- If the word before was either "to" or "by" then
    - If there is no object or if the current verb is not an "action" verb then current subject will be made the subject of the modifying phrase.
    - Otherwise the current object will be made the subject of the modifying phrase.

- If the previous word was neither "to" nor "by" then
    - If the grammar of the current post-modifier of the object is either a noun or a pronoun, then make the noun or pronoun the subject of the modifying phrase.
    - Otherwise the current subject will be the subject of the modifying phrase.
- Increment the current definition pointer by one and go to the next indicated state.

State Q46 - *Verb or Auxiliary - Create a new phrase*

- Create a new phrase.
- Add the phrase new the current phrase section.
- If there is an object make it the subject of the new phrase, otherwise use the current subject of the previous phrase.
- Increment the current definition pointer by one and go to the next indicated state.

State Q47 - *Pronoun - Create a new phrase*

- Create a new phrase.
- Add the phrase new the current phrase section.
- If the count of the current object post-modifier is compatible with that of the pronoun and the grammar of the current object post-modifier is a noun then make the current object post-modifier the subject of the newly created phrase.
- Otherwise, increment the current definition pointer by one and go to the next indicated state.

State Q48 - *Adjective*

- Add the adjective as a pre-modifier of the action.
- Increment the current definition pointer by one and go to the next indicated state.

State Q49 - *Verb*

- Add the verb as a post-modifier of the action.
- Increment the current definition pointer by one and go to the next indicated state.

State Q50 - *Comma or Conjunction*

- If the following phrase is not in the form of "conjunction adverb conjunction" (i.e. *as heavy as*) then
  - Create a new phrase node.
  - Go to state Q0.
- Otherwise, increment the current definition pointer by one and go to the next indicated state.

Automaton Data Flows

Figure C.1 - Subject Automation with Pre-Modifier States

Figure C.2 - Subject Automaton with Post-Modifier States

Figure C.3 - Action Automaton with Pre-Modifier and Post-Modifier States

Figure C.4 - Object Automaton with Pre-Modifier States

Figure C.5 - Object Automaton with Post-Modifier States

Appendix D: Dictionary Seed

```
Definition Structure

   text#0
      XXX#0
      noun#0
         proper#0
            name#0
            personal name#0
            geographical name#0
               landmark#0
               land mass#0
               population center#0
               land division#0
               water body#0
               watercourse#0
            name + common noun#0
               roadway#0
               person#0
            temporal name#0
               festival#0
               duration#0
               time#0
         common#0
            inanimate#0
               partitive#0
                  typical#0
                  measure#0
                     speed#0
                     duration#1
                     volume#0
                     weight#0
                     distance#0
                     time#1
                     temperature#0
                     loudness#0
                     texture#0
                     quality#0
                     emotion#0
                     personality#0
                     direction#0
                     size#0
                     shape#0
                     color#0
                     age#0
                     degree#0
                     number#0
                     appeal#0
                     appearance#0
               style#0
               provenance#0
               concrete#0
                  location#3
                  organization#0
```

```
            structure#0
            container#0
            object#0
            element#0
            roadway#1
            geography#0
               landmark#1
               population center#1
               land division#1
               water body#1
               land mass#1
               watercourse#1
         abstract#0
            object#1
            communication#0
            action#0
            activity#0
            cognition#0
            process#0
            relational#0
            sensation#0
            stative#0
            transitional#0
      animate#0
         creature#0
            human#0
            animal#0
            insect#0
            part#0
         plant#0
            part#1
verb#0
   action#1
   auxiliary#1
      have#4
         have#5
         have#6
         have#7
      be#2
         be#3
         are#4
      do#0
   activity#1
   cognition#1
      call#0
         called#0
         calling#0
   process#1
   relational#1
      are#0
   sensation#1
   stative#1
   transitional#1
auxiliary#0
   primary#0
      have#0
         have#1
         have#2
```

```
         have#3
      be#0
         be#1
         are#1
         are#2
         are#3
   modal#0
adjective#0
   gradable#0
      speed#1
      quantity#0
      duration#2
      volume#1
      weight#1
      distance#1
      temperature#1
      loudness#1
      texture#1
      time#2
      location#0
      quality#1
      emotion#1
      personality#1
      direction#1
      size#1
      degree#1
      shape#1
      color#1
      age#1
      number#1
      appeal#1
      appearance#1
   non-gradable#0
      stative#2
      style#1
      provenance#1
      action#2
      activity#2
      cognition#2
      process#2
      relational#2
      sensation#2
      transitional#2
determiner#0
   their#0
   this#0
premodifier#0
   quantifier#0
      both#0
      all#0
   ordinal#0
   cardinal number#0
   predeterminer#0
adverb#0
   quantity#1
   manner#1
   time#3
   location#2
```

```
   sentential#2
   degree#2
   speed#2
   duration#3
   distance#2
   temperature#2
   loudness#2
   texture#2
   time#4
   quality#2
   emotion#2
   personality#2
   direction#2
   size#2
   shape#2
   appeal#2
   appearance#2
pronoun#0
   there#0
preposition#0
   to#0
   of#0
   by#0
conjunction#0
   coordinate#0
   correlative#0
      or#0
   subordinate#0
interjection#0
question#0
negative#0
syntax#0
   period#0
   comma#0
   question#1
```

Appendix E: System Defined Relations

HAS SUBSET - Parent to child relation indicating child is a subset of the parent entity.

HAVE - Parent to child relation indicating the parent has a connected body part referenced by the child entity.

includes - Parent to child minor relation indicating child is a member of the parent entity.

INCLUDES - Parent to child major relation indicating child is a member of the parent entity.

instance - Parent to child minor relation indicating child is an instance of the parent entity.

INSTANCE - Parent to child major relation indicating child is an instance of the parent entity.

instance of - Child to parent minor relation indicating child is an instance of the parent entity.

INSTANCE OF - Child to parent major relation indicating child is an instance of the parent entity.

IS - Parent to child relation indicating the parent has an attribute referenced by the child entity.

isa - Child to parent minor relation indicating child is a member of the parent entity.

ISA - Child to parent major relation indicating child is a member of the parent entity.

ISNOTA - Child to parent relation indicating child is not a member of the parent entity.

SUBSET OF - Child to parent relation indicating child is a subset of the parent entity.

Appendix F: System Logic Tests

Nineteen sets of sentences were processed by the machine conceptualization process to test the machine's logic capabilities when formulating its semantic net. The results of these nineteen tests are listed in this appendix.

Example: 1

Elephants are gray.
Royal elephants are elephants.
Royal elephants are not gray.
Clyde is a royal elephant.

```
entity [1] <1> (0)
   elephant#0 [1] <1> (6)
      royal elephant#0 [1] <1> (8)
         Clyde#0 [1] <1> (3)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=elephant#0

Entity = elephant#0
   Relations :
       0.0  A=elephant#0  R=ISA  O=entity
       0.0  A=elephant#0  R=IS  O=gray#0
       1.0  A=elephant#0  R=INCLUDES  O=royal elephant#0 Quantity=4

Entity = royal elephant#0
   Relations :
       1.0  A=royal elephant#0  R=ISA  O=elephant#0 Quantity=4
       2.0  A=royal elephant#0  R=ISNOT  O=gray#0
       3.1  A=royal elephant#0  R=INSTANCE  O=Clyde#0

Entity = Clyde#0
   Relations :
       3.1  A=Clyde#0  R=INSTANCE OF  O=royal elephant#0
```

Example: 2

Elephants are gray.
Royal elephants are elephants.
Royal elephants are not gray.
Clyde is a royal elephant.
Clyde is an elephant.

```
entity [1] <1> (0)
   elephant#0 [1] <1> (6)
      royal elephant#0 [1] <1> (8)
         Clyde#0 [1] <1> (3)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=elephant#0

Entity = elephant#0
   Relations :
       0.0  A=elephant#0  R=ISA  O=entity
       0.0  A=elephant#0  R=IS  O=gray#0
       1.0  A=elephant#0  R=INCLUDES  O=royal elephant#0 Quantity=4

Entity = royal elephant#0
   Relations :
       1.0  A=royal elephant#0  R=ISA  O=elephant#0 Quantity=4
       2.0  A=royal elephant#0  R=ISNOT  O=gray#0
       3.1  A=royal elephant#0  R=INSTANCE  O=Clyde#0

Entity = Clyde#0
   Relations :
       3.1  A=Clyde#0  R=INSTANCE OF  O=royal elephant#0
```

Example: 3

Elephants are gray.
Gray things are drab.
Elephants are drab.
Royal elephants are elephants.
Royal elephants are not gray.

```
entity [1] <1> (0)
   elephant#0 [1] <1> (6)
      royal elephant#0 [1] <1> (7)
   thing#0 [2] <1> (2)
      gray#0 thing#0 [1] <1> (3)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=elephant#0
       1.0  A=entity  R=INCLUDES  O=thing#0

Entity = elephant#0
   Relations :
       0.0  A=elephant#0  R=ISA  O=entity
       0.0  A=elephant#0  R=IS  O=gray#0
       2.0  A=elephant#0  R=IS  O=drab#0
       3.0  A=elephant#0  R=INCLUDES  O=royal elephant#0 Quantity=3

Entity = royal elephant#0
       3.0  A=royal elephant#0  R=ISA  O=elephant#0 Quantity=3
       4.0  A=royal elephant#0  R=ISNOT  O=gray#0

Entity = thing#0
   Relations :
       1.0  A=thing#0  R=ISA  O=entity
       1.0  A=thing#0  R=HAS SUBSET gray#0  O=thing#0

Entity = gray#0 thing#0
   Relations :
       1.0 gray#0  A=thing#0  R=SUBSET OF  O=thing#0
       1.0 gray#0  A=thing#0  R=IS  O=drab#0
```

Example: 4

Quakers are pacifists.
Republicans are not pacifists.
Nixon is a Quaker.
Nixon is a Republican.

```
entity [1] <1> (0)
   pacifist#0 [1] <1> (2)
      quaker#0 [1] <1> (3)
         Nixon#0 [1] <1> (3)
   republican#0 [1] <1> (3)
      Nixon#0 [1] <1> (3)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=pacifist#0
       1.0  A=entity  R=INCLUDES  O=republican#0

Entity = pacifist#0
   Relations :
       0.0  A=pacifist#0  R=ISA  O=entity
       0.0  A=pacifist#0  R=INCLUDES  O=quaker#0

Entity = quaker#0
   Relations :
       0.0  A=quaker#0  R=ISA  O=pacifist#0
       2.1  A=quaker#0  R=INSTANCE  O=Nixon#0

Entity = Nixon#0
   Relations :
       2.1  A=Nixon#0  R=INSTANCE OF  O=quaker#0
       3.0  A=Nixon#0  R=instance of  O=republican#0

Entity = republican#0
   Relations :
       1.0  A=republican#0  R=ISA  O=entity
       1.0  A=republican#0  R=ISNOTA  O=pacifist#0
       3.0  A=republican#0  R=instance  O=Nixon#0
```

Example: 5

Elephants are shy.
Performers are not shy.
Circus performers are performers.
Clyde is a circus performer.
Clyde is an elephant.

```
entity [1] <1> (0)
   elephant#0 [1] <1> (3)
      Clyde#0 [1] <1> (3)
   performer#0 [1] <1> (3)
      circus#1 performer#0 [1] <1> (4)
         Clyde#0 [1] <1> (3)
   circus#0 [3] <1> (1)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=elephant#0
       1.0  A=entity  R=INCLUDES  O=performer#0
       3.0  A=entity  R=INCLUDES  O=circus#0

Entity = elephant#0
   Relations :
       0.0  A=elephant#0  R=ISA  O=entity
       0.0  A=elephant#0  R=IS  O=shy#0
       4.0  A=elephant#0  R=instance  O=Clyde#0

Entity = performer#0
   Relations :
       1.0  A=performer#0  R=ISA  O=entity
       1.0  A=performer#0  R=ISNOT  O=shy#0
       2.0  A=performer#0  R=HAS SUBSET circus#1  O=performer#0

Entity = circus#1 performer#0
   Relations :
       2.0 circus#1  A=performer#0  R=SUBSET OF  O=performer#0
       3.1 circus#1  A=performer#0  R=INSTANCE  O=Clyde#0

Entity = Clyde#0
   Relations :
       3.1  A=Clyde#0  R=INSTANCE OF circus#1  O=performer#0
       4.0  A=Clyde#0  R=instance of  O=elephant#0

Entity = circus#0
   Relations :
       3.0  A=circus#0  R=ISA  O=entity
```

Example: 6

Birds can fly.
The ostrich is a bird.
The ostrich can not fly.

```
entity [1] <1> (0)
   bird#0 [1] <1> (3)
      ostrich#0 [1] <1> (4)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=bird#0

Entity = bird#0
   Relations :
       0.0  A=bird#0  R=ISA  O=entity
       0.0  A=bird#0  R=can#0  O=fly#3
       1.0  A=bird#0  R=INCLUDES  O=ostrich#0

Entity = ostrich#0
   Relations :
       1.0  A=ostrich#0  R=ISA  O=bird#0
       2.0  A=ostrich#0  R=can#0 not#0  O=fly#3
```

Example: 7

Birds can fly.
The ostrich is a bird.
The ostrich can not fly.
Henry is an ostrich.
Henry can fly.

```
entity [1] <1> (0)
   bird#0 [1] <1> (3)
      ostrich#0 [1] <1> (5)
         Henry#0 [1] <1> (4)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=bird#0

Entity = bird#0
   Relations :
       0.0  A=bird#0  R=ISA  O=entity
       0.0  A=bird#0  R=can#0  O=fly#3
       1.0  A=bird#0  R=INCLUDES  O=ostrich#0

Entity = ostrich#0
   Relations :
       1.0  A=ostrich#0  R=ISA  O=bird#0
       2.0  A=ostrich#0  R=can#0 not#0  O=fly#3
       3.1  A=ostrich#0  R=INSTANCE  O=Henry#0

Entity = Henry#0
   Relations :
       3.1  A=Henry#0  R=INSTANCE OF  O=ostrich#0
       4.0  A=Henry#0  R=can#0  O=fly#3
```

Example: 8

All Quakers are Christians.
Many Quakers dislike churches.

```
entity [1] <1> (0)
   Christian#0 [1] <1> (2)
      quaker#0 [1] <1> (3)
   church#0 [2] <1> (1)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=Christian#0
       1.0  A=entity  R=INCLUDES  O=church#0

Entity = Christian#0
   Relations :
       0.0  A=Christian#0  R=ISA  O=entity
       0.0  A=Christian#0  R=INCLUDES  O=quaker#0

Entity = quaker#0
   Relations :
       0.0  A=quaker#0  R=ISA  O=Christian#0
       1.0  A=quaker#0  R=HAS SUBSET many#0  O=quaker#0

Entity = many#0 quaker#0
   Relations :
       1.0 many#0  A=quaker#0  R=SUBSET OF  O=quaker#0
       1.0 many#0  A=quaker#0  R=dislike#1  O=church#0

Entity = church#0
   Relations :
       1.0  A=church#0  R=ISA  O=entity
```

Example: 9

Many Quakers are teetotalers.
Many teetotalers are Quakers.

```
entity [1] <1> (0)
   quaker#0 [1] <1> (3)
   teetotaler#0 [1] <1> (3)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=quaker#0
       0.0  A=entity  R=INCLUDES  O=teetotaler#0

Entity = quaker#0
   Relations :
       0.0  A=quaker#0  R=ISA  O=entity
       0.0  A=quaker#0  R=HAS SUBSET many#0  O=quaker#0
       1.0  A=quaker#0  R=INCLUDES many#0  O=teetotaler#0

Entity = many#0 quaker#0
   Relations :
       0.0 many#0  A=quaker#0  R=SUBSET OF  O=quaker#0
       0.0 many#0  A=quaker#0  R=ISA  O=teetotaler#0

Entity = many#0 teetotaler#0
   Relations :
       1.0 many#0  A=teetotaler#0  R=SUBSET OF  O=teetotaler#0
       1.0 many#0  A=teetotaler#0  R=ISA  O=quaker#0

Entity = teetotaler#0
   Relations :
       0.0  A=teetotaler#0  R=ISA  O=entity
       0.0  A=teetotaler#0  R=INCLUDES many#0  O=quaker#0
       1.0  A=teetotaler#0  R=HAS SUBSET many#0  O=teetotaler#0
```

Example: 10

Joe is a human.
Larry is a human.
Humans have two legs.
Joe has two legs.
Larry has two legs.

```
entity [1] <1> (0)
   human#0 [1] <1> (6)
      Joe#0 [1] <1> (3)
      Larry#0 [1] <1> (3)
   leg#0 [2] <1> (2)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=human#0
       2.0  A=entity  R=INCLUDES  O=leg#0

Entity = human#0
   Relations :
       0.0  A=human#0  R=ISA  O=entity
       0.1  A=human#0  R=INSTANCE  O=Joe#0
       1.1  A=human#0  R=INSTANCE  O=Larry#0
       2.0  A=human#0  R=HAVE two#0  O=leg#0 Quantity=3

Entity = Joe#0
   Relations :
       0.1  A=Joe#0  R=INSTANCE OF  O=human#0

Entity = Larry#0
   Relations :
       1.1  A=Larry#0  R=INSTANCE OF  O=human#0

Entity = leg#0
   Relations :
       2.0  A=leg#0  R=ISA  O=entity
       2.0  A=leg#0  R=HAS SUBSET two#0  O=leg#0

Entity = two#0 leg#0
   Relations :
       2.0 two#0  A=leg#0  R=SUBSET OF  O=leg#0 Quantity=3
```

Example: 11

John is human.
Humans have two legs.
John has one leg.

```
entity [1] <1> (0)
   human#0 [2] <1> (3)
      John#0 [1] <1> (4)
   leg#0 [1] <1> (3)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=human#0
       1.0  A=entity  R=INCLUDES  O=leg#0

Entity = human#0
   Relations :
       0.0  A=human#0  R=ISA  O=entity
       0.1  A=human#0  R=INSTANCE  O=John#0
       1.0  A=human#0  R=HAVE two#0  O=leg#0

Entity = John#0
   Relations :
       0.1  A=John#0  R=INSTANCE OF  O=human#0
       2.0  A=John#0  R=HAVE one#1  O=leg#0

Entity = leg#0
   Relations :
       1.0  A=leg#0  R=ISA  O=entity
       1.0  A=leg#0  R=HAS SUBSET two#0  O=leg#0
       2.0  A=leg#0  R=HAS SUBSET one#1  O=leg#0

Entity = two#0 leg#0
   Relations :
       1.0 two#0  A=leg#0  R=SUBSET OF  O=leg#0

Entity = one#1 leg#0
   Relations :
       2.0 one#1  A=leg#0  R=SUBSET OF  O=leg#0
```

Example: 12

People are persons.
Persons are people.*

* Nothing to show, because the above is a circular definition.

Example: 13

Whales are mammals.
Some whales have teeth.
Dolphins are mammals.
Some dolphins do not have teeth.
Porpoises are mammals.
Porpoises have teeth.

```
entity [1] <1> (0)
   mammal#0 [1] <1> (4)
      whale#0 [1] <1> (3)
         toothed#0 whale#0 [1] <1> (3)
      dolphin#0 [1] <1> (3)
         toothless#0 dolphin#0 [1] <1> (3)
      porpoise#0 [3] <1> (3)
   tooth#0 [2] <1> (1)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=mammal#0
       1.0  A=entity  R=INCLUDES  O=tooth#0

Entity = mammal#0
   Relations :
       0.0  A=mammal#0  R=ISA  O=entity
       0.0  A=mammal#0  R=INCLUDES  O=whale#0
       2.0  A=mammal#0  R=INCLUDES  O=dolphin#0
       4.0  A=mammal#0  R=INCLUDES  O=porpoise#0

Entity = whale#0
   Relations :
       0.0  A=whale#0  R=ISA  O=mammal#0
       1.0  A=whale#0  R=HAS SUBSET toothed#0  O=whale#0

Entity = toothed#0 whale#0
   Relations :
       1.0 toothed#0  A=whale#0  R=SUBSET OF  O=whale#0
       1.0 toothed#0  A=whale#0  R=HAVE  O=tooth#0

Entity = dolphin#0
   Relations :
       2.0  A=dolphin#0  R=ISA  O=mammal#0
       3.0  A=dolphin#0  R=HAS SUBSET toothless#0  O=dolphin#0

Entity = toothless#0 dolphin#0
   Relations :
       3.0 toothless#0  A=dolphin#0  R=SUBSET OF  O=dolphin#0
       3.0 toothless#0  A=dolphin#0  R=HAVENOT  O=tooth#0

Entity = porpoise#0
   Relations :
       4.0  A=porpoise#0  R=ISA  O=mammal#0
       5.0  A=porpoise#0  R=HAVE  O=tooth#0

Entity = tooth#0
   Relations :
       1.0  A=tooth#0  R=ISA  O=entity
```

Example: 14

Smokey is a brown bear.
Smokey is a mammal.
Bears are mammals.

```
entity [1] <1> (0)
   mammal#0 [1] <1> (2)
      bear#0 [1] <1> (3)
         brown bear#0 [1] <1> (3)
            Smokey#0 [1] <1> (2)

Entity = entity
   Relations :
       1.0  A=entity  R=INCLUDES  O=mammal#0

Entity = mammal#0
   Relations :
       1.0  A=mammal#0  R=ISA  O=entity
       2.0  A=mammal#0  R=INCLUDES  O=bear#0

Entity = bear#0
   Relations :
       2.0  A=bear#0  R=ISA  O=mammal#0
       3.0  A=bear#0  R=INCLUDES  O=brown bear#0

Entity = brown bear#0
   Relations :
       0.1  A=brown bear#0  R=INSTANCE  O=Smokey#0
       3.0  A=brown bear#0  R=ISA  O=bear#0

Entity = Smokey#0
   Relations :
       0.1  A=Smokey#0  R=INSTANCE OF  O=brown bear#0
```

Example: 15

John is a tarantula.
Tarantula is an arachnid.
Spiders are arachnids.
Tarantula is a spider.

```
entity [1] <1> (0)
   arachnid#0 [1] <1> (2)
      spider#0 [1] <1> (3)
         tarantula#0 [1] <1> (3)
            John#0 [1] <1> (2)

Entity = entity
   Relations :
       1.0  A=entity  R=INCLUDES  O=arachnid#0

Entity = arachnid#0
   Relations :
       1.0  A=arachnid#0  R=ISA  O=entity
       2.0  A=arachnid#0  R=INCLUDES  O=spider#0

Entity = spider#0
   Relations :
       2.0  A=spider#0  R=ISA  O=arachnid#0
       3.2  A=spider#0  R=INCLUDES  O=tarantula#0

Entity = tarantula#0
   Relations :
       0.1  A=tarantula#0  R=INSTANCE  O=John#0
       3.2  A=tarantula#0  R=ISA  O=spider#0

Entity = John#0
   Relations :
       0.1  A=John#0  R=INSTANCE OF  O=tarantula#0
```

Example: 16

Gray elephants are elephants.

```
entity [1] <1> (0)
   elephant#0 [1] <1> (2)
      gray#0 elephant#0 [1] <1> (2)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=elephant#0

Entity = elephant#0
   Relations :
       0.0  A=elephant#0  R=ISA  O=entity
       0.0  A=elephant#0  R=HAS SUBSET gray#0  O=elephant#0

Entity = gray#0 elephant#0
   Relations :
       0.0 gray#0  A=elephant#0  R=SUBSET OF  O=elephant#0
```

Example: 17

Rhinos are black or white.
Zebras are black and white.
Elephants are performers or shy.

```
entity [1] <1> (0)
   rhino#0 [3] <1> (2)
   zebra#0 [1] <1> (3)
   performer#0 [1] <1> (2)
      elephant#0 [1] <1> (3)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=rhino#0
       1.0  A=entity  R=INCLUDES  O=zebra#0
       2.0  A=entity  R=INCLUDES  O=performer#0

Entity = rhino#0
   Relations :
       0.0  A=rhino#0  R=ISA  O=entity
       0.0  A=rhino#0  R=IS black#0 or#0  O=white#0

Entity = zebra#0
   Relations :
       1.0  A=zebra#0  R=ISA  O=entity
       1.0  A=zebra#0  R=IS  O=black#0
       1.0  A=zebra#0  R=IS  O=white#0

Entity = performer#0
   Relations :
       2.0  A=performer#0  R=ISA  O=entity
       2.0  A=performer#0  R=INCLUDES  O=elephant#0

Entity = elephant#0
   Relations :
       2.0  A=elephant#0  R=ISA  O=performer#0
       2.0  A=elephant#0  R=IS or#0  O=shy#0
```

Example: 18

Elephants are Republicans or performers.

```
entity [1] <1> (0)
   republican#0 [1] <1> (2)
      elephant#0 [1] <1> (3)
   performer#0 [1] <1> (2)
      elephant#0 [1] <1> (3)

Entity = entity
   Relations :
       0.0  A=entity  R=INCLUDES  O=republican#0
       0.0  A=entity  R=INCLUDES  O=performer#0

Entity = republican#0
   Relations :
       0.0  A=republican#0  R=ISA  O=entity
       0.0  A=republican#0  R=INCLUDES  O=elephant#0

Entity = elephant#0
   Relations :
       0.0  A=elephant#0  R=ISA  O=republican#0
       0.0  A=elephant#0  R=isa or#0  O=performer#0

Entity = performer#0
   Relations :
       0.0  A=performer#0  R=ISA  O=entity
       0.0  A=performer#0  R=includes  O=elephant#0
```

Example: 19

Clyde is an elephant.
Clyde is a circus elephant.
Circus Elephant is a royal elephant.
Royal elephant is an elephant.
Royal elephant is not a gray thing.
Elephant is a gray thing.
Gray thing is a drab thing.

```
entity [1] <1> (0)
   circus#0 [2] <1> (1)
   thing#0 [1] <1> (3)
      gray#0 thing#0 [1] <1> (4)
         elephant#0 [1] <1> (8)
            elephant#0 [1] <1> (4)
               Clyde#0 [1] <1> (3)
            royal elephant#0 [1] <1> (7)
               elephant#0 [1] <1> (4)
                  Clyde#0 [1] <1> (3)
      drab#0 thing#0 [2] <1> (3)
         gray#0 thing#0 [1] <1> (4)
            elephant#0 [1] <1> (8)
               elephant#0 [1] <1> (4)
                  Clyde#0 [1] <1> (3)
               royal elephant#0 [1] <1> (7)
                  elephant#0 [1] <1> (4)
                     Clyde#0 [1] <1> (3)

Entity = entity
   Relations :
       1.0  A=entity  R=INCLUDES  O=circus#0
       4.0  A=entity  R=INCLUDES  O=thing#0

Entity = circus#0
   Relations :
       1.0  A=circus#0  R=ISA  O=entity

Entity = thing#0
   Relations :
       4.0  A=thing#0  R=ISA  O=entity
       4.0  A=thing#0  R=HAS SUBSET gray#0  O=thing#0
       6.0  A=thing#0  R=HAS SUBSET drab#0  O=thing#0

Entity = gray#0 thing#0
   Relations :
       4.0 gray#0  A=thing#0  R=SUBSET OF  O=thing#0
       6.0 gray#0  A=thing#0  R=isa drab#0  O=thing#0
       5.2 gray#0  A=thing#0  R=INCLUDES  O=elephant#0

Entity = elephant#0
   Relations :
       1.0  A=elephant#0  R=HAS SUBSET circus#1  O=elephant#0
       3.0  A=elephant#0  R=INCLUDES  O=royal elephant#0 Quantity=4
       5.2  A=elephant#0  R=ISA gray#0  O=thing#0

Entity = circus#1 elephant#0
   Relations :
       1.0 circus#1  A=elephant#0  R=SUBSET OF  O=elephant#0
       1.1 circus#1  A=elephant#0  R=INSTANCE  O=Clyde#0
       2.0 circus#1  A=elephant#0  R=ISA  O=royal elephant#0
```

```
Entity = Clyde#0
   Relations :
       1.1  A=Clyde#0  R=INSTANCE OF circus#1  O=elephant#0

Entity = royal elephant#0
   Relations :
       2.0  A=royal elephant#0  R=INCLUDES circus#1  O=elephant#0
       3.0  A=royal elephant#0  R=ISA  O=elephant#0 Quantity=4
       4.0  A=royal elephant#0  R=ISNOTA gray#0  O=thing#0

Entity = drab#0 thing#0
   Relations :
       6.0 drab#0  A=thing#0  R=SUBSET OF  O=thing#0
       6.0 drab#0  A=thing#0  R=includes gray#0  O=thing#0
```

Appendix G: System Referenced Words

all#0 - Premodifier acting as a quantifier.

are#3 - Auxiliary verb in the present tense.

be#2 - Verb acting as an auxiliary in the present tense.

both#0 - Premodifier acting as a quantifier.

by#0 - Preposition.

called#0 - Preposition.

comma#0 - Syntax.

coordinate#0 - Conjunction.

do#0 - Verb acting as an auxiliary.

have#4 - Verb acting as an auxiliary.

of#0 - Preposition.

or#0 - Correlative conjunction.

part#0 - Noun representing an animate entity body part.

text#0 - Root node of the dictionary, its grammar is undefined.

their#0 - Determiner that shows possession.

there#0 - Pronoun.

this#0 - Determiner.

to#0 - Preposition.

XXX#0 - Generic system variable, grammar is undefined.

Appendix H: Program Data Structures

```
/*********************************************/
/* Define The String Object                  */
/*********************************************/
class String
{
private:
    char   *buf;          //string location (string pointer)
    short  length;        //length of string
};

/***********************************/
/* class ParsedText                */
/***********************************/
class ParsedText
{
private:
    String **wrd_ptr;
    int    wrd_qty;
};

/*********************************************/
/* Define The Attribute Table Element Object  */
/*********************************************/
class AttrElement
{
private:
    String *name;         //Attribute Name
    int    index;         //Attribute reference index
    String **value;       //Attribute's possible values
    int    val_qty;       //Quantity of possible attribute values
};

/***********************************/
/* The Attribute Table Object      */
/***********************************/
class AttrTable
{
private:
    AttrElement    **tbl_ent;     //Attribute Table Entries
    int            ent_qty;       //Quantity of Attribute Table Entries
};

/***********************************/
/* class Attribute                 */
/***********************************/
class Attribute
{
private:
    Attribute  *nxt_ptr;       //Point to the next attribute
    int        att_ind;        //Attribute Table index
    int        val_ind;        //Attribute Table value index
};
```

```
/************************************/
/* Definition Structure             */
/************************************/
class Definition
{
private:
    Definition **incl_ptr;       //Includes Pointer
    Definition *isa_ptr;         //ISA Pointer
    LexEntry   *lex_ptr;         //Points to the natural word
    Attribute  *att_ptr;         //Points to the attributes
    String     *def_wrd;         //Definition word
    short      def_lvl;          //Definition level
    short      incl_qty;         //Includes Pointer Quantity
    short      def_flag;         //Definition Flag
    short      grmr_ind;         //Indicates to Grammar Type
};

/************************************/
/* class LexEntry                   */
/************************************/
class LexEntry
{
private:
    LexEntry   **lex_ptr;     //Pointer to an array of lexicon entries
    Definition **def_ptr;     //Pointer to lexicon definitions
    String     *lex_word;     //Lexicon Natural Word
    int        lex_qty;       //Number of connected lexicon entries
    int        def_qty;       //Number of definitions for word
};

/************************************/
/* class Lexicon                    */
/************************************/
class Lexicon
{
private:
    LexEntry   **lexicon;     //Pointer to the lexicon entries
};

/************************************/
/* class Sentence                   */
/************************************/
class Sentence
{
private:
    Definition     **wrd_ptr;     //Word Pointers
    int        wrd_qty;        //Word Quantity
};

/************************************/
/* class Paragraph                  */
/************************************/
class Paragraph
{
private:
    Sentence   **sen_ptr;        //Sentence Pointers
    int        sen_qty;          //Sentence Quantity
};
```

```
/************************************/
/* class Book                       */
/************************************/
class Book
{
private:
    Paragraph  **par_ptr;        //Paragraph Pointers
    int        par_qty;          //Paragraph Quantity
};

/***************************************/
/* Subject Object                      */
/***************************************/
class Subject
{
private:
    Definition *sub;             //Subject
    Definition **pre_mod;        //Pre-Modifiers
    Definition **post_mod;       //Post-Modifiers
    Concept    *mod_con;         //Modifying Concepts
    int        pre_qty;          //Pre-Modifier Quantity
    int        post_qty;         //Post-Modifier Quantity
};

/***************************************/
/* Action Object                       */
/***************************************/
class Action
{
private:
    Definition *act;          //Action
    Definition **pre_mod;     //Pre-Modifiers
    Definition **post_mod;    //Post-Modifiers
    Subject    **obj;         //Object of Action
    int        pre_qty;       //Pre-Modifier Quantity
    int        post_qty;      //Post-Modifier Quantity
    int        obj_qty;       //Object Quantity
};

/***************************************/
/* Concept Object - Really a Phrase    */
/***************************************/
class Concept
{
private:
    Subject    **sub_ptr;        //Subject Pointers
    Action     **act_ptr;        //Action Pointers
    int        sub_qty;          //Subject Quantity
    int        act_qty;          //Action Quantity
    int        sen_nbr;          //Originating Sentence Number
};
```

```
/*******************************************************/
/* Concept Section Object - Really a group of phrases */
/*******************************************************/
class ConceptSection
{
private:
    Concept **c_ptr;   //Pointer to a group of phrases
    int     c_qty;     //Quantity of phrases in a group
};

/*************************************************************/
/* Concept Root Object - Really a grouping of Phrase groups */
/*************************************************************/
class ConceptRoot
{
private:
    ConceptSection **c_ptr;   //Pointer to groups of phrases
    int            c_qty;     //Quantity of phrase groupings
};

/*************************************/
/* Relation Object                   */
/*************************************/
class Relation
{
private:
    String     *name;         //relation name
    String     **rel_pre;     //relation pre-modifier names
    int        rpre;          //relation pre-modifier quantity
    String     **rel_post;    //relation post-modifier names
    int        rpost;         //relation post-modifier quantity
    String     **sub_pre;     //subject pre-modifier names
    int        spre;          //subject pre-modifier quantity
    String     **sub_post;    //subject post-modifier names
    int        spost;         //subject post-modifier quantity
    String     **obj_pre;     //object pre-modifier names
    int        opre;          //object pre-modifier quantity
    String     **obj_post;    //object post-modifier names
    int        opost;         //object post-modifier quantity
    float      con_nbr;       //originating concept structure number
    int        rel_qty;       //relationship quantity
    int        neg_sw;        //Indicates if relationship is negative
    int        moved;         //Indicates relation has moved up the net
};
```

```
/*****************************************/
/* Entity Object                         */
/*****************************************/
class Entity
{
private:
    String     *name;         //entity name
    int        ent_qty;       //individual relations going out from entity
    int        out_qty;       //total relations going out from entity
    int        in_qty;        //relations coming into entity
    Relation   **rel_ptr;     //entity relations with other entities
    Entity     **ent_ptr;     //other entities, entity has relations with
    int        distance;      //distance from root entity
    int        out_up;        //ent_qty + parent's ent_qty
    int        out_down;      //accumulation of children's ent_qty
    int        exp_rt;        //best example rating
    int        exp_pts;       //best example points
    int        imp_rt;        //importance rating
    int        imp_pts;       //importance points
    int        bas_pts;       //basic-level points
    int        inst_sw;       //SUBSET OF switch
    int        prnt_sw;       //indicates if entity has been printed
    int        prc_sw;        //indicates if entity has been processed
};

/*****************************************/
/* Words Object                          */
/*****************************************/
class Words
{
private:
    String  *wrd;             //pointer to a word
    Words   *nxt;             //next word
    Words   *sen;             //next sentence
    Words   *par;             //next paragraph
};
```

Appendix I: Questionnaire Results

Comments

- In questions one through three (*Most Knowledgeable About, Best Examples, Most Important*) the entities could receive a ranking from one to seven. A rank of one means the "*most*" or the "*best*" while a rank of seven the "*least*" or the "*worst*".

- In questions one and three the following machine importance method labels have the following meanings:
  - ***m91*** - *Emanating Arcs* method
  - ***m92*** - *Entering Arcs* method
  - ***m93*** - *Arcs In & Out* method
  - ***m94*** - *All Descendants* method
  - ***m97*** - *All Ancestors* method

- In question two the following machine *best example* method labels have the following meanings:
  - ***m2*** - *Minimal* method
  - ***m3*** - *Combination* method
  - ***m4*** - *Multiplier* method
  - ***m5*** - *Multiplier & Combination* method

Bears - Most Knowledgeable About

Whales - Most Knowledgeable About

Spiders - Most Knowledgeable About

Bears - Best Example

Whales - Best Example

Spiders - Best Example

Bears - Most Important

Whales - Most Important

Spiders - Most Important

Bears - Categorization

Whales - Categorization

Spiders - Categorization

Bears - Basic Level Entities

Whales - Basic Level Entities

Spiders - Basic Level Entities

Appendix J: Questionnaire Statistics

Comments

- In questions one and three the following machine importance method labels have the following meanings:
  - ***m91*** - *Emanating Arcs* method
  - ***m92*** - *Entering Arcs* method
  - ***m93*** - *Arcs In & Out* method
  - ***m94*** - *All Descendants* method
  - ***m97*** - *All Ancestors* method

- In question two the following machine *best example* method labels have the following meanings:
  - ***m2*** - *Minimal* method
  - ***m3*** - *Combination* method
  - ***m4*** - *Multiplier* method
  - ***m5*** - *Multiplier & Combination* method

Bears - Most Knowledgeable About - Rank Group

Whales - Most Knowledgeable About - Rank Group

Spiders - Most Knowledgeable About - Rank Group

Bears - Most Knowledgeable About - Rank Individual

Whales - Most Knowledgeable About - Rank Individual

Spiders - Most Knowledgeable About - Rank Individual

Bears - Best Example - Rank Group

Whales - Best Example - Rank Group

Spiders - Best Example - Rank Group

Bears - Best Example - Rank Individual

Whales - Best Example - Rank Individual

Spiders - Best Example - Rank Individual

Bears - Most Important - Rank Group

Whales - Most Important - Rank Group

Spiders - Most Important - Rank Group

Bears - Most Important - Rank Individual

Whales - Most Important - Rank Individual

Spiders - Most Important - Rank Individual

Bears - Categorization

Whales - Categorization

Spiders - Categorization

Bears - Basic Level Entities

Whales - Basic Level Entities

Spiders - Basic Level Entities

Appendix K: Methodology Table

Class	Name	Method	Reference
Classification	Incremental Classification "ID5R"	**Representation**: Decision Tree **Purpose**: Create a decision procedure for classifying instances into categories. (incremental) **Algorithm**: " 1. If the tree is empty, then define it as the unexpanded form, setting the class name to the class of the instance, and the set of instances to the singleton set containing the instance. 2. Otherwise, If the tree is in unexpanded form and the instance is from the same class, then add the instance to the set of instances kept in the node. 3. Otherwise, (a) If the tree is in unexpanded form, then expand it one level, choosing the test attribute for the root arbitrarily. (b) For the test attribute and all non-test attributes at the current node, update the count of positive or negative instances for the value of that attribute in the training instance. (c) If the current node contains an attribute test that does not have the lowest E-score, then i. Restructure the tree so that an attribute with the lowest E-score is at the root ii. Recursively reestablish a best test attribute in each subtree except the one that will be updated in step 3d. (d) Recursively update the decision tree below the current decision node along the branch for the value of the test attribute that occurs in the instance description. Grow the branch if necessary."	[Utgoff, 1989, pp. 165-167] [Ragavan, Rendell, Shaw, Tessmer, 1993, pp. 946-947]
Classification	Incremental Classification Tree-Update for ID4	**Representation**: Decision Tree **Purpose**: Create a decision procedure for classifying instances into categories. **Algorithm**: 1. Add the instance at the current node. 2. If all the instances at the node are homogeneous then classify node as either a "yes" or "no" node. 3. Otherwise: (a) If the current node is a leaf node, calculate entropy with the remaining attributes and branch on the attribute with the lowest entropy. (b) Otherwise: (i) Calculate entropy with the remaining attributes. (ii) If current branching attribute does not have the lowest entropy, discard all the existing subtrees below the current node, branch on the attribute with the lowest entropy, and then rebuild the tree below the current node minimizing entropy as you create new branches. (iii) Otherwise continue down the tree.	[Utgoff, 1989, p. 164]

Classification	Instance Partitioning Classification	**Representation**: Hierarchical Structure **Purpose**: "Builds a tree of categories by recursively partitioning instances into disjoint sets, and classifies instances by sorting along a single path in the tree." **Algorithm**: Initialization: 1) Create a root category and add the first instance to it. When a new instance is encountered do the following: 2) Initialize Best. 3) Do the following for each node in the hierarchy. 3a) If the node is a leaf node: i) Temporarily create a new node with the instance. ii) Create a new node whose children are the instance node and the current node. 3b) If the node is not a leaf node: i) For each subcategory the current node, temporarily add the instance to it and calculate the predictive accuracy. ii) Temporarily create a new category and estimate its predictive accuracy. iii) If the predictive accuracy is better than Best, set Best equal to the current predictive accuracy. 4) Permanently add the instance to the best scoring subcategory.	[Martin, Billman, 1991, pp. 80-87]
Classification	Non-Incremental Classification "ID3"	**Representation**: Decision Tree **Purpose**: Create a decision tree using a sample set of objects. Their attributes will determine where branches are to be created. **Algorithm**: Initialization: 1) Create a root node and put the entire sample set in it. For each leaf in the tree do the following: 2) If all the objects in the leaf are positive make the leaf a "yes" node. 3) If all the objects in the leaf are negative make the leaf a "no" node. 4) If the leaf has mixed set of positive and negative objects do the following: 4a) Use a heuristic to select one of the remaining attributes that will minimize entropy. 4b) Create a branch and leaf node from the current leaf node for each possible value of the selected attribute. 4c) Classify each object in the current node by its value for the selected attribute.	[Quinlan, 1986, pp. 88-90] [Thorton, 1992, pp. 72-78] [Utgoff, 1989, pp. 162-163]

Classification	Non-Incremental Classification	**Representation**: Decision Tree **Purpose**: Create a decision tree using a sample set of objects. Their attributes will determine where branches are to be created. **Algorithm**: Initialization: 1) Create a root node and put the entire sample set in it. For each leaf in the tree do the following: 2) If all the objects in the leaf are positive make the leaf a "yes" node. 3) If all the objects in the leaf are negative make the leaf a "no" node. 4) If the leaf has mixed set of positive and negative objects do the following: 4a) Randomly select an attribute that hasn't been used. 4b) Create a branch and leaf node from the current leaf node for each possible value of the selected attribute. 4c) Classify each object in the current node by its value for the selected attribute.	[Thorton, 1992, p. 72]
Classification	Pattern Composition Classification	**Representation**: Hierarchical structure **Purpose**: To add instances to a hierarchical structure by putting them in current nodes or creating new ones.. **Algorithm**: Initialization: 1) Create a root category and add the first instance to it. When a new instance is encountered do the following: 2) For the current category in the hierarchy do the following: 2a) If the category is a leaf node add a new leaf node using the instance and return. 2b) Otherwise, for each subcategory do the following: 2bi) Add the instance to the subcategory and calculate its predictive accuracy. 2bii) Create a temporary category with the instance and calculate its predictive accuracy. 2c) Permanently add the instance to the most predictive subcategory. 3) Make the most predictive category the current category and go to (2).	[Martin, Billman, 1991, pp. 81-87]
Clustering	Agglomerative Hierarchical Clustering	**Representation** : Dendrogram/Numerical Taxonomy **Purpose** : Create a dendrogram using some distance method to cluster individuals together using a bottom-up strategy. **Algorithm** : Initialize : Create n clusters, each cluster containing a single individual. Choose a distance algorithm to compute the distance between clusters. 1. Compute the distance between all the individual clusters. 2. Choose the pair of clusters that are the nearest (Ci and Cj) and merge them into cluster Ci and delete cluster Cj. 3. Decrement the number of clusters by 1. 4. If the number of clusters is equal to 1 stop. 5. Return to 1.	[Everitt, 1993, pp. 56-57] [Fisher & Pazzani, 1991, pp. 12-13]

Clustering	Centroid Clustering	**Representation** : Dendrogram **Purpose** : Create a dendrogram using some distance method to cluster individuals together using a bottom-up strategy. **Algorithm** : *Initialize* : Create n clusters, each cluster containing a single individual. *Distance Method* - With this method, groups once formed are represented by their mean values for each variable, that is, their mean vector, and inter group distance is now defined in terms of distance between two such mean vectors. Possible group means: Euclidean Distance : City Block : 1. Compute the distance between all the individual clusters. 2. Choose the pair of clusters that are the nearest (Ci and Cj) and merge them into cluster Ci and delete cluster Cj. 3. Decrement the number of clusters by 1. 4. If the number of clusters is equal to 1 stop. 5. Return to 1.	[Everitt, 1993, pp. 62-64]
Clustering	Conceptual Clustering	**Representation** : Semantic Net Rules **Purpose**: Create intermediate concepts which compress raw knowledge, fill in incompleteness, and decrease quantity of computations. **Algorithm**: One-Step procedure "Input : rule system R. Step 1: Calculate the distance between each pair of rules in R. Step 2: Using each rule as a seed, collect all rules within a distance t1 from the seed and form a star. Step 3: Calculate the relative frequency for each condition and each action occurring in each star. Step 4: For a representative rule for each star. The condition set contains all conditions with relative frequencies of occurrences in the start greater than t2. The action set contains all action with relative frequencies greater than t3. Step 5: Replace each seed with the representative rule for the star centering at it and call the new rule system R'." **Algorithm**: One-Layer procedure "Input : rule system R. Step 1: Apply one-step conceptual clustering to R to get R'. Step 2: If the number of distinct rules in R' is equal to that in R, go to step 3. Otherwise, let R = R' and got to step 1. Step 3: Delete redundant rules in R' and form S (A rule system containing representative rules for one layer concepts)."	[Cheng & Fu, 1985, p. 594]

Clustering	Entity-Relationship Clustering	**Representation**: Entity-Relationship Model
		Algorithm:
		For each relationship set (RSS) do
		If RSS order > 2 introduce associative RSS.
		If m-n relationship introduce associative RSS.
		If 1-1 RSSs of type ((1:1) and (1:1)) or ((0:1) and (0:1))
		perform clustering immediately.
		
		If no root entity exit algorithm.
		Else
		Build hierarchy of entity sets (ESs).
		If cycle detected then exit cycle
		level = order of hierarchy just built
		While level >= 2 do
		Find all clustering candidates (CCs) on level
		For each CC do
		found-existential-parent (found-EP) = False
		For each parent entity (PE) set do
		add CC to PE set
		If CC participates in recursive RSS Then
		establish recursive RSS involving PE set
		If found-EP = False Then check for EP
		If EP Then
		found-EP = True
		Fix EP
		End For
		If found-EP
		For Each PE set of CC do
		Create/update complex relationship (CR) between PE set and on EP.
		For each CR where CC is do
		Substitute EP for CC
		Else
		For each PE set of CC do
		Create/update relationship between PE sets
		For each complex relationship where CC is Do
		Substitute PE set for CC
		level = level - 1
		End While
			[Rauh & Stickel, 1992, pp. 73-75]

Clustering	Furthest Neighbor Clustering	**Representation** : Dendrogram **Purpose** : Create a dendrogram using some distance method to cluster individuals together using a bottom-up strategy. **Algorithm** : *Initialize* : Create n clusters, each cluster containing a single individual. *Distance Method* - is the opposite of the nearest neighbor clustering in the sense that distance between groups is now defined as that of the most distant pair of individuals, one from each group. 1. Compute the distance between all the individual clusters. 2. Choose the pair of clusters that are the nearest (Ci and Cj) and merge them into cluster Ci and delete cluster Cj. 3. Decrement the number of clusters by 1. 4. If the number of clusters is equal to 1 stop. 5. Return to 1.	[Everitt, 1993, pp. 60-61]

Clustering	Group Averaging Clustering	**Representation** : Dendrogram **Purpose** : Create a dendrogram using some distance method to cluster individuals together using a bottom-up strategy. **Algorithm** : *Initialize* : Create n clusters, each cluster containing a single individual. *Distance Method* - The distance between two clusters is defined as the average of the distances between all pairs of individuals that are made up of one individual from each group. 1. Compute the distance between all the individual clusters. 2. Choose the pair of clusters that are the nearest (Ci and Cj) and merge them into cluster Ci and delete cluster Cj. 3. Decrement the number of clusters by 1. 4. If the number of clusters is equal to 1 stop. 5. Return to 1.	[Everitt, 1993, pp. 61-62]
Clustering	Mode Analysis Clustering	**Representation** : Dendrogram **Purpose** : Create a dendrogram using some distance method to cluster individuals together using a bottom-up strategy. **Algorithm** : " The search is made by considering a sphere of some radius R, surrounding each point and counting the number of points falling in the sphere. Individuals are then labeled as dense or non-dense depending on whether their spheres contain more or less points than the value of the linkage parameter , K, which is preset at a value dependent on the number of individuals in the data set. The parameter R is gradually increased and so more individuals became 'dense'. Four courses of action are possible with the introduction of each new dense point. (i) The new point is separated from all other dense points by a distance which exceeds R. when this happens the point initiates a new cluster nucleus and the number of clusters i s increased by one. (ii) The new point is within distance R of one or more dense points which belong to only one cluster nucleus. In this case, the new point is added to the existing cluster. (iii) The new point is within distance R of dense points belong to two or more clusters. If this happens the clusters concerned are combined. (iv) At each 'introduction' cycle the smallest distance, D, between dense points belong to different clusters found, and compared with a threshold value calculated from the average of the 2K smallest distance coefficients for each individual. if D is less than this threshold value then the two clusters are combined. sometimes only one cluster is produced Indicating a lack of cluster structure in the data), but usually the analysis reaches a point at which a maximum number of clusters is isolated. it is usually this solution which is taken as the most significant."	[Everitt, 1993, p. 127]

Clustering	Nearest Neighbor Clustering	**Representation** : Dendrogram **Purpose** : Create a dendrogram using some distance method to cluster individuals together using a bottom-up strategy. **Algorithm** : *Initialize* : Create n clusters, each cluster containing a single individual. *Distance Method* - is that distance between groups is defined as that of the closest pair of individuals, where only pairs consisting of one individual from each group are considered. 1. Compute the distance between all the individual clusters. 2. Choose the pair of clusters that are the nearest (Ci and Cj) and merge them into cluster Ci and delete cluster Cj. 3. Decrement the number of clusters by 1. 4. If the number of clusters is equal to 1 stop. 5. Return to 1.	[Everitt, 1993, pp. 57-60]
Clustering	Unsupervised Clustering "EPAM"	**Representation**: Decision Tree **Purpose**: Models rote learning behavior, word recognition, and chess patterns. **Algorithm**:" A new observation is classified down a path of matching arcs to a leaf. If the observation matches the concept stored at the leaf, then a process of familiarization occurs: the leaf is specialized by adding a feature that is present in the observation and not present in the current leaf. If there is a mismatch of one or more features, then one of two actions is taken. The observation and pattern may disagree on a dimension corresponding to an 'else' link that was taken during classification. If so, then this point of disagreement is found and two new arcs are created, one with the value of the new observation and one with the value of the leaf. The latter arc leads to the same subtree as before. The former arc leads to a new leaf that is constructed to fit the new observation, thus increasing the breadth of the tree. This leaf contains all of the features of the observation that were used in test during classification. If no disagreement along the classification path is found, then the difference that triggered the original search is used to differentiate the leaf's pattern and a new leaf corresponding to the observation, thus increasing the depth of the tree. Again, this pattern contains the features on the path from root to leaf. In sum, leaves begin as general patterns that are gradually specialized by familiarization as subsequent observations are classified."	[Fisher & Pazzani, 1991, pp. 23-24]

Clustering	Unsupervised Clustering "CYRUS", "UNIMEM"	**Representation**: Decision Tree **Purpose**: Create a decision tree using a top-down strategy. **Algorithm**:" First, to actually be classified at a node, the observation must match the concept description stored at the node. ... In both systems, an observation is classified under a node if it shares a specified proportion of the node's predictable attribute values. During the top-down classification process, if no child of the current node sufficiently matches the observation, then the observation is made a new child of the node. To enforce the predictable status of attribute values, each value is weighted. This is an integer weight; whenever an observation is compared against a candidate concept, an attribute value's weight is incremented if the observation and node agree along this attribute value's weight is decremented in the case of disagreement. If the weight falls below a user-defined threshold, then the attribute value is dropped from the concept description. when a new node is created, the predictable values of the new node are those values of the observation that are not predictable values of the new node's parent. Thus, values that are shared by the observation and parent are inherited from the node's parent."	[Fisher & Pazzani, 1991, pp. 24-25]
Clustering	Unsupervised Clustering "COBWEB", "OXBOW"	**Representation**: Dendrogram **Purpose**: Create a dendrogram incrementally. **Algorithm**: Initialization: Create a root node. For every new instance do the following, starting from the root node: 1. If current node is the root node update all the base rates for each node stored at the root node. 2. Temporarily add the instance to the current node and calculate the conditional probabilities (category utility) for each child. 3. Do one of the following using the result from (2). 3a. Classify the instance down into the next level. 3b. If the current node has no subclasses: i. Create a new class. ii. Attach the instance and the current node to it. iii. Return. 3c. Create a new subclass using the instance and attach it to the current node. 3d. Merge the best two subclasses and classify the instance into the new subclass. 3e. Split the best subclass and go to 2.	[Fisher & Pazzani, 1991, p. 27] [Iba & Gennari, 1991, pp. 367-371] [Reich & Fenves, 1991, pp. 334-335]
Clustering Measure	Inverse of Euclidean Distance	**Representation**: Numerical Taxonomy **Purpose**: "measure the similarity between observations" **Method**: " , where x_{ik} and x_{jk} are the values of attribute k for observation i and j, respectively; the closer the values of individual attributes, the smaller the denominator of the inverted distance, and the greater the similarity."	[Fisher & Pazzani, 1991, p. 13]

Conceptual Graph Utility	Type Contraction	**Representation**: Conceptual Graph
		Purpose: "Type contraction deletes a complete subgraph and incorporates the equivalent information in the type label of a single concept."
		Condition: "Let u be a canonical graph, and let type t be defined as λav. If u is a specialization of v, π is a projection v into u, and type(πa) = type(a), then the operation of type contraction may be performed on u..."
		Algorithm:
		Replace the type label of πa with t.
		Leave referent(πa) unchanged;
		For b in the concepts and conceptual relation of v where b ≠a, πb identical to b, and πb not a cut point of u loop
		If b is a concept then
		detach πb from u;
		Else
		detach πb and all its arcs from u;
		End loop
		For e in the arcs left in u not linked to a concept loop
		reattach the concept that had been linked to arc e in u;
		End loop;
			[Sowa, 1984, p. 107-108]

Conceptual Graph Construction	Perception	**Representation**: Conceptual Graph **Algorithm**: "The process of perception generates a structure u called a conceptual graph in response to some external entity or scene e: - The entity e gives rise to a sensory icon s. - The associative comparator finds one or more percepts $p_1,...,p_n$ that match all or part of s. - The assembler combines the percepts $p_1,...,p_n$ to form a working model that approximates s. - If such a working model can be constructed, the entity e is said to be recognized by the percepts $p_1,...,p_n$. - For each percept p_i in the working model, there is a concept c_i called the interpretation of p_i. - T_e concepts $c_1,...,c_n$ are linked by conceptual relations to form the conceptual graph u."	[Sowa, 1984, p. 70]
Conceptual Graph Utility	Canonical Indicator	**Representation**: Conceptual Graph **Purpose**: Indicates if the conceptual graph G is canonical. **Algorithm**:" Canonical(well-formed graph G, canonical basis β) For all r-vertex r of G, Mark(r) ← False While there is an r of G with Mark(r) = False For all $B_i \in \beta$ If Projection(B_i, G, r, Mark) Then exit (for loops) If Mark(r) = False Then return false. Return True. End Projection(T, G, r_g, Mark) For all r_i in T such that label(r_t) = label(r_g) If PROJ(r_t, r_g, Mark) Then return True Return False End"	[Mugnier & Chein, 1993, p. 311]

Conceptual Graph Utility	Verb Instance Derivation	**Representation**: Conceptual Graph
		Purpose: To retrieve and interpret verb instances and interpret verb instances to be used as input for learning algorithms to generate verb taxonomies.
		Algorithm:
		Let Cat = category, R = relation, esl = elementary syntactive link, CF = certainty factory, SM_v = Semantic Modifiers of v.
		Let S be a sentence of the corpus and S' the set of elementary syntactic links.
		1) Select the position, pos(v) of a verb v in S. let esl(v,prep,n) $\in$ S' be the corresponding syntactic links, validated by R_i,such as Cat - (R_i) - Cat_j / prep, with v $\in$ Cat (generally Cat=Act) and n $\in Cat_j$.
		2)
		If esl(v,prep,n) has a lexical representation of v Then
		If CF(esl(v,prep,n)) = 1 Then
		ADD - (R_i) - Cat_j to the list of SM_v
		Else
		If esl(v(prep,n) is ambiguous respect esl(n',prep,n)
		If prob(esl(v,prep,n)) > prob(esl(n',prep,n) Then
		ADD - (Rj) - Cat to the list of SM_v
		Else
		leave esl(v,prep,n) unanalyzed
		Else
		If prob(esl(v,prep,n)) > α (α=0.2) Then
		ADD - R_j - Cat_j to the list of SM_v
		Else
		leave esl(v,prep,n) unanalyzed.
		3) Repeat step 1 and 2 fore each esl(v,prep',n') $\in$ S'.
		4) Repeat steps 1 - 3 for each verbal link remaining in S'.
			[Basili & Pazienza, 1993, pp. 166 - 177]

Conceptual Graph Utility	Well-Formed Indicator	**Representation**: Conceptual Graph **Purpose**: Indicates if a conceptual graph G is well-formed relative to the support S where S provides background knowledge on a specific domain application. Well-formed means if a graph conforms to the constraints derived from S. **Algorithm**: " Well-Formed (conceptual graph G, support S) For all r-vertex r Let tr = lab(r) and β_{tr} be the star graph associated with tr. If degree(r) in G $\neq$ edge number of B_{tr}. Then return false For all i $\in$ {i, ..., degree(r)} Let c be the ith neighbor of r' in G. Let t_i be the label of the ith neighbor of the r-vertex in B_{tr} If not(lab(c) $\leq$ (t_i, *)) The return false. Return true End"	[Mugnier & Chein, 1993, p. 297]
Deduction	Learning by Deduction	The learner acquires a concept by deducing it from the knowledge given and/or possessed.	[R. S. Michalski, 1992, p 250]

ER Utility	Schema Integration	**Representation**: Entity Relationship Model **Purpose** : "To create a global schema, to provide a common access or interface to local (or component) schemata." **Algorithm**: " 1) In the canonizing step, the intra-schema conflicts and inconsistencies are detected against some predefined criteria or requirements, such as 'an entity type is not allowed to connect to an attribute by a relationship.' 2) The comparison step performs a pair wise comparison of objects of the schemata to be integrated and finds possible object pairs which may be semantically similar. The object pair set so generated is called the semantic similarity relation with respect to some properties, such as synonym, equal key attributes and equal context. 3) In the conciliation step, a variety of user assisted techniques are used to resolve conflicts and mismatched objects. 4) The merging step generates an integrated schema from two component schemata. 5) The last step, restructuring, the objective is to check the consistency of the integrated schema and build correspondences between the component schemata and the integrated schema."	[Song, Johannesson, Bubenko, 1992, pp. 97-98]
ER Utility	Valence Expression Interpretation	**Representation:** Entity-Relationship Models **Definition**: Valence - "A valence is a binary, directed edge emanating from a relationship an directed towards another object(entity or relationship)." **Algorithm**:" 1) Every type identifier T is an expression. The interrelation is I(T) = [T]. The following rule serves to connect expression by and, xor, and or further, parentheses are introduced. 2) If A and B are expressions, then: A and B, A xor B, A or B, (A) are also expressions. They are interpreted according to the following rules: $I(A \text{ and } B) = \{M \mid M = M_A \cup M_B, M_A \in I(A), M_B \in I(B)\}$ $I(A \text{ xor } B) = I(A) \cup I(B)$ I(A or B) = I(A xor B xor (A and B)) I((A)) = I(A). The following rules define cardinality constraints. 3) Let A be a expression and m a natural number. Then m A is an expression. The interpretation is defined recursively as: I (0 A) = [] I(m A) = I(m' A and A) with m' = m -1 for m > 0. 4) Let m and n be natural number with m < n. Then, m .. n A is an expression. It is to be interpreted as: I(m .. m A) = I(m A) I(m .. n A) = I(m .. n' A xor n A) with n' = n -1 for m < n. 5) For m ≥ 0, m ..* A, is an expression. The interpretation is defined as: I(m.. *A) = I(m A and 0 ..*A) for m > 0."	[Zhou & Baumann, 1992, pp. 27-29]

ER Utility	Valence Expression Normalizatio n	**Representation:** Entity-Relationship Model **Algorithm**:" 1) For a type identifier T, its normal for is T. 2) Suppose A and B are two normalized sub expressions. a) For the valence expression A and B, we normalize it using the distributivity of and over xor to 'draw out' xor in each subexpression. b) For the valence expression A xor B, we normalize it by merging the same terms in A and B. c) For the valence expression A or B, we normalize its equivalent form A xor B xor (A and B) using the rules 2(a) and 2(b). d) For the valence expression (A), its normal form is A. 3) Suppose A is a normalized subexpression and m and n are two natural number with m ≤ n. a) For the valence express m A, we first expand m A to a sequence of M occurrences of A connected by and. Then, we normalize it using the rule 2(a). b) For the valence expression m..n A, we first break m..n A down into alternatives whose cardinality specifiers run through all numbers between m and n. Then, we normalize the valence expression representing each alternative using the rule 3(a) and normalize the entire expression using the rule 2(b). c) For the valence expressions m..* A, which is equivalent to m A and 0..* A, we first normalize the subexpression 0..* a, Using Theorem 0..* (A xor B) = 0..* A and)..* B, we distribute '0..*' over xor. Using the theorems: 0..* (0..* A) = 0..* A , and 0..*(A and 0..*B) = 0..* A and 0..*B, we eliminate the nested '0..*'. After merging the same terms, factors, and the atom factors having the same type identifiers, we obtain the normal form for o..* A. Then, using the rule 2(a) and 3(a), we normalize the entire expression."	[Zhou, Baumann, 1992, pp. 32-33]
Goal Search	A* Search	**Representation** : Search Tree **Purpose** : A branch-and-bound search, with an estimate of remaining distance, combined with the dynamic-programming principle. **Algorithm** : " - Form a one-element queue consisting of a zero-length path that contains only the root node. - Until the first path in the queue terminates at the goal node or the queue is empty, - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node. - Reject all new paths with loops. - Add the remaining new paths, if any, to the queue. - If two or more paths reach a common node, delete al those paths except the one that reaches the common node with the minimum cost. - Sort the entire queue by the sum of the path length and a lower-bound estimate of the cost remaining, with least-cost paths in front. - If the goal node is found, announce success; otherwise, announce failure."	[Winston, 1992, p. 94]

Goal Search	Alpha-Beta (Depth-First Search)	**Representation** : Two player game trees
		Purpose : Find the best path through the tree structure.
		Algorithm : "
		function FAB(Position, Alpha, Beta) : integer;
		M := SelectNextMove(Position);
		if M = null the return (StaticEvaluation(Position));
		Best := -∞;
		while M ≠ null do
		Value := - FAB(M(Position), -Beta, -max(Alpha, Best));
		if Value > Best then
		Best := Value;
		if Best ≥ Beta then return(Best)
		end if
		M := SelectNextMove(Position);
		end while
		return (Best);
		end function
			[Kaindl, 1990, p. 138]

Goal Search	Branch-and - Bound Search	**Representation** : Search Tree **Purpose** : Find a least-cost partial path through a tree to reach a goal. **Algorithm** : " - Form a one-element queue consisting of a zero-length path that contains only the root node. - Until the first path in the queue terminates at the goal node or the queue is empty, - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node. - Reject all new paths with loops. - Add the remaining new paths, if any, to the queue. - Sort the new paths, if any, by the estimated distances between their terminal nodes and the goal. - If the goal node is found, announce success; otherwise, announce failure."	[Winston, 1992, p. 83]
Goal Search	Branch-and - Bound Search with dynamic programming	**Representation** : Search Tree **Purpose** : Find a path through a tree to reach a goal using the philosophy "the best way through a particular, intermediate place is the way to it from the starting place, followed by the best way from it to the goal". **Algorithm** : " - Form a one-element queue consisting of a zero-length path that contains only the root node. - Until the first path in the queue terminates at the goal node or the queue is empty, - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node. - Reject all new paths with loops. - Add the remaining new paths, if any, to the queue. - If two or more paths reach a common node, delete al those paths except the one that reaches the common node with the minimum cost. - Sort the entire queue by path length with least-cost paths in front. - If the goal node is found, announce success; otherwise, announce failure."	[Winston, 1992, p. 91]

Goal Search	Branch-and - Bound Search with lower bound estimate	**Representation** : Search Tree **Purpose** : Find a least-cost partial path through a tree to reach a goal. **Algorithm** : " - Form a one-element queue consisting of a zero-length path that contains only the root node. - Until the first path in the queue terminates at the goal node or the queue is empty, - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node. - Reject all new paths with loops. - Add the remaining new paths, if any, to the queue. - Sort the entire queue by the sum of the path length and a lower-bound estimate of the cost remaining, with the least-cost paths in front. - If the goal node is found, announce success; otherwise, announce failure."	[Winston, 1992, p. 88]
Goal Search	Breadth-First Search	**Representation** : Search Tree **Purpose** : Find a path through a tree to reach a goal. **Algorithm** : " - Form a one-element queue consisting of a zero-length path that contains only the root node. - Until the first path in the queue terminates at the goal node or the queue is empty, - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node. - Reject all new paths with loops. - Add the new paths, if any, to the back of the queue. - If the goal node is found, announce success; otherwise, announce failure."	[Winston, 1992, p.68]
Goal Search	Depth-First Search	**Representation** : Search Tree **Purpose** : Find a path through a tree to reach a goal. **Algorithm** : " - Form a one-element queue consisting of a zero-length path that contains only the root node. - Until the first path in the queue terminates at the goal node or the queue is empty, - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node. - Reject all new paths with loops. - Add the new paths, if any, to the front of the queue. - If the goal node is found, announce success; otherwise, announce failure."	[Winston, 1992, p.68]

Goal Search	Hill-Climbing Search	**Representation** : Search Tree **Purpose** : Find a path through a tree using a heuristic to reach a goal. **Algorithm** : " - Form a one-element queue consisting of a zero-length path that contains only the root node. - Until the first path in the queue terminates at the goal node or the queue is empty, - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node. - Reject all new paths with loops. - Sort the new paths, if any, by the estimated distances between their terminal nodes and the goal. - Add the new paths, if any, to the front of the queue. - If the goal node is found, announce success; otherwise, announce failure."	[Winston, 1992, p.70]
Goal Search	Non-Deterministic Search	**Representation** : Search Tree **Purpose** : Find a path randomly through a tree to reach a goal. **Algorithm** : " - Form a one-element queue consisting of a zero-length path that contains only the root node. - Until the first path in the queue terminates at the goal node or the queue is empty, - Remove the first path from the queue; create new paths by extending the first path to all the neighbors of the terminal node. - Reject all new paths with loops. - Add the new paths at random places in the queue. - If the goal node is found, announce success; otherwise, announce failure."	[Winston, 1992, p.69]
Hierarchical Classification	Feature Comparison Method	**Representation**: Hierarchy **Purpose**: Categorizes concepts within a hierarchy according the number of common features and feature weights of the item and target concepts. **Algorithm**: Given target and test concepts do the following: 1) Count the number of feature matches between the target and test concepts. 2) If the number of matches is greater than some maximum threshold classify the test concept with the target concept and return. 3) If the number of matches is less than some minimum threshold retrieve another target concept and goto (1). 4) If the number of matches is between the minimum and maximum thresholds do the following: 4a) If all the high weighted features of the target concept match all the high weighted features of the test concept add the test concept to the target concept. 4b) Otherwise retrieve another target concept and goto (1).	[Smith, Medin, 1981, p. 80]

Hierarchical Classification	Spreading Activation Method	**Representation** : Hierarchy **Purpose**: Categorizes concepts within a hierarchy according the number of common and non-common features and with their corresponding importance. **Algorithm**: Given a target concept and a test concept do the following: 1) Activate all the target concept's features. 2) Activate all the test concept's features. 3) When ever a target and test feature intersect do the following: 3a) Increment a positive evidence accumulator. 3b) If the positive accumulator exceeds some threshold classify the test concept with the concept and return. 4) When ever a target and test feature don't intersect do the following: 4a) Increment a negative evidence accumulator. 4b) If the negative accumulator exceeds some threshold retrieve another target and goto (1).	[Smith, Medin, 1981, pp. 76-77]
Hierarchical Utility	Category Utility	**Representation**: Hierarchical Structure **Purpose**: Measure basic level effects. **Method**: "... category utility measure, which assumes that basic level concepts maximize : where $P(C_k)$ is the proportion of a population covered by a category, C_k; $P(V_j\|C_k)$ is the category validity or predictability of value Vj relative to C_k; and $P(V_j)$ is the base rate probability of B_j. Category utility is a tradeoff between the degree that C_k increases the expected number of correct predictions about the presence of attribute value over the uninformed case and the proportion of the population to which this increase applies given by the $P(C_k)$ term."	[Fisher & Pazzani, 1991, p. 26]
Inheritance	Backward Chaining Inheritance	**Representation**: Hierarchies **Algorithm**: "The rule is that $<x_1, ..., x_n>$ is inheritable in Φ if and only if Φ contains $<x_1, x_2>$ and $<x_2, ..., x_n>$, and Φ neither contradicts nor precludes $<x_1, ..., x_n>$."	[Touretzky, 1986, p. 68]
Inheritance	Computing Semantic Distance	**Representation**: Semantic Net **Algorithm**: To find the semantic distance between two concepts C1 with type T1 and C2 with type T2, find the concept C3 which generalizes C1 and C2 with type T3 such that T3 is the most specified type which subsumes T1 and T2; the semantic distance between C1 and C2 is the sum of the distances from C1 to C3 and C2 to C3. This is a modification of Sowa's metric.	[Foo, 1992, pp. 149-154]

Inheritance	Double Chaining Inheritance	**Representation**: Hierarchies **Algorithm**: "A sequence $\sigma = <x_1, ..., x_n>$ is inheritable in Φ if and only if $n > 2$, Φ contains both $<x_1, ..., x_{n-1}>$ and $<x_2, ..., x_n>$, and Φ neither contradicts nor precludes σ." **Definition**: Precludes "Φ precludes a sequence $r = <x_1, ..., x_n>$ iff Φ contains a sequence $<y,x_n'>$ where y is an intermediary to σ in Φ." **Definition**: Contradicts "A set of sequences Φ contradicts the sequence $<x_1, ..., x_n>$ iff $<x_1, x_i'> C(\Phi)$ for some i, $1 \leq i \leq n$."	[Touretzky, 1986, p. 43]
Inheritance	Forward Chaining Inheritance	**Representation**: Hierarchies **Algorithm**: "Our rule is that $<x_1, ..., x_n>$ is inheritable in Φ if and only if Φ contains $<x_1, ... x_{n-1}>$ and $<x_{n-1}, ..., x_n>$, and Φ neither contradict nor precludes $<x_1, ..., x_n>$."	[Touretzky, 1986, p. 66]
Inheritance	Inferential Distance	**Representation**: Multiple Inheritance Hierarchies **Algorithm**: "The essential intuition behind inheritance exceptions is: subclasses override super classes. Briefly stated, the inferential distance rule says that A may view B as a subclass of C iff A has an inference path vie B to C, and not vice versa. Suppose we are trying to find out if A has property P. If A inherits from B which has property P, and also from C which has property ~P, what conclusion would we reach about A? The inferential distance ordering say: if A has an inference path via B to C and not vice versa, then conclude that A has P; if A has an inference path via C to B and not vice versa, then conclude that A has ~P; otherwise there is an ambiguity."	[Touretzky, 1986, p. 208]

Inheritance	Preemptive Strategy (Specificity)	**Knowledge Representation**: Inheritance Hierarchy
		Purpose: Chooses the shortest path to reach the goal node.
		Algorithm: "
		Let Γ be an inheritance hierarchy.
		Sort the nodes of Γ topologically.
		For each focus node a;
		$\Sigma = \Gamma$;
		For each node x reachable from a in Σ, in topological order;
		For each edge $v \cdot (\sim)x \in E\Sigma$, in reverse topological order;
		Let $\Gamma^* = \Sigma - \{q \mid q$ is a preemptor of $v \cdot (\sim)x$ in $\Sigma\}$
		If Γ^* no longer contains a positive path from a to v then
		remove the edge $v \cdot (\sim)x$ from Σ.
		For each remaining positive edge $p \cdot x$;
		If Σ contains a path $p \cdot q_1 \cdot \cdot \cdot q_n \cdot x$ $(n \geq 1)$ such that there is no negative path through admissible edges from a to any of q_i, x in Σ then
		mark the edge $p \cdot x$ as redundant."
			[Stein, 1991, pp. 277-278]

Learning	Direct Implanting of Knowledge	This is an extreme case in which the learner does not have to perform any inference on the information provided.	[R. S. Michalski, 1992, p. 249]

Learning	Learning by Analogy	The learner acquires a new concept by modifying the definition of a known similar concept.	[R. S. Michalski, 1992, p. 250]
Learning	Learning by Instruction	Here the learner acquires concepts from a teacher or other organized source, such as a publication or textbook, but does not directly copy into memory the information supplied.	[R. S. Michalski, 1992, pp. 249-250]
Learning	Learning by Observation and Discovery	In this strategy the learner analyzes given and/or observed entities and determines that some subsets of these entities can be grouped usefully into certain classes (i.e. concepts).	[R. S. Michalski, 1992, pp. 250-251]
Learning from Experience	Chunking	"A chunk is a unit of memory organization, formed by bringing together a set of already formed chunks in memory and welding them together into a large unit. Chunking implies the ability to build up such structures recursively, thus leading to a hierarchical organization of memory. Chunking is learning from experience. It is a way of converting goal-based problem solving into accessible long-term memory (production). Whenever problem solving has provided some result, a new production will be created, whose actions are these just obtained results and whose conditions are the working-memory elements that existed before the problem solving started that were used to produce the results. This newly minted production will be added to the long-term memory, and will hence forth be available to add its knowledge to the working memory, and will henceforth be available to add its knowledge to the working memory in any future elaboration phase where it conditions are satisfied."	[Newell, 1990, p. 7]
Learning from Experience	Learning from Experience	The learner induces a concept description by generalizing from teacher or environment provided examples and (optionally) counter examples.	[R. S. Michalski, 1992, p. 250]
Parsing	Metaphor Recognition	**Representation**: Conceptual Graphs **Algorithm**: "1) First try to analyze each sentence literally. If no canonical graph can be formed for a grammatical sentence, consider it as a possible metaphor. 2) Check the catalog of common analogies to find a possible transfer of schema for one or more of the concepts in the sentence. 3) Determine whether a conceptual graph that represents the sentence can be canonically derived from the transferred schema. 4) If a suitable interpretation is found, remember the schema that was transferred, since it is likely to be used again within the same story or conversation."	[Carbonell, 1982] in [Sowa, 1984, p. 271]
Parsing	Predictive Grammar Parser	The Predictive Grammar Parser determines the correct sense of each word in a sentence using a predictive grammar table and a prediction pool with prediction sub-pools. The predictive grammar table contains predictions with their associate syntactic word classes and syntactic role indicators. A word's syntax is used as key into the predictive table and a set of syntax predictions are returned in the form of predictive sub-pools for each possible future syntax of the next word. Impossible combinations are eliminated as well as predictions that don't come true. As the end of the sentence is reached several versions of the sentence are produced with the different senses of each word.	[S. Kuno, S. and A. G. Oettinger, 1962.]

Partitioning	Bayesian Analysis	**Representation**: Probabilistic Category Model **Purpose**: Categorization **Algorithm**: For every object encountered: " 1. If no previous object has been seen, initialize the category partitioning of the objects to be the empty set. 2. Given a partitioning for the first m object, calculate for each category k the probability P_k that the m +1st object comes from category k. Let P_0 be the probability that the object comes from a completely new category. 3. Create a partitioning of the m+1 objects in which the m+1st object is assigned to the category with maximum probability. 4. To predict value j on dimension i for the n+1st object calculate , where P_k is the probability that the n+1st object comes from category k and P(ij|k) is the probability of displaying value j on dimension i given membership in k."	[Anderson & Matessa, 1991, p. 47]
Partitioning	Bayesian Analysis Incremental	**Representation**: Hierarchy Structure **Purpose**: Build a hierarchical organization. **Algorithm**: " 1. Given a new instance, determine a category K to associate with this instance. 2. If K is an existing category, sort the new instance to a location below that category node. 3. If K is a new category, sort that category to a location below the top node for the hierarchy, with the constraint that one category cannot be placed under anther. 4. Search upward from where the new item was inserted to see whether some change in the category structure is warranted. Note that this does no reorganize the hierarchy, but only changes which nodes in the hierarchy might be considered category nodes."	[Anderson & Matessa, 1991, p. 62]

Rule Chaining	Backward Chaining	**Representation** : Rule base **Purpose** : Moves from initial hypothesis, through rules to known facts, establishing variable bindings in the process to come up with a solution. **Algorithm** : " - Find a rule whose consequent matches the hypothesis (or antecedent) and create a binding set (or augment the existing binding set). - Using the existing binding set, look for a way to deal with the first antecedent, - Try to match the antecedent with an existing assertion. - treat the antecedent as an hypotheses and try to support it by backward chaining through other rules using the existing binding set. - Repeat the previous step for each antecedent, accumulating variable bindings, until, - There is no match with any existing assertion or rule consequent using the binding set established so far. In this case, back up to the most recent match with unexplored bindings, looking for an alternative match that produces a workable binding set. - There are no more antecedents to be matched. In this case, the binding set in hand supports the original hypothesis. - If all possible binding sets are desired, report the current binding set, and back up, as if there were no match. - If only one possible binding set is desired, report the current binding set and quit. -There are no more alternative matches to be explored at any level.	[Winston, 1992, p. 147]

Rule Chaining	Forward Chaining	**Representation :** Rule base **Purpose** : Discovers solutions by making assertions. **Algorithm** : " - Until no rule produces a new assertion, - For each rule, - Try to match the first antecedent with and existing assertion. Create a new binding set with variable bindings established by the match. - Using the existing variable bindings, try to match the next antecedent with an existing assertion. If any new variables appear in this antecedent, augment the existing variable bindings. - Repeat the previous step for each antecedent, accumulating variable bindings as you go , until, - There is no match with any existing assertion using the binding set established so far. In this case, back up to a previous match of an antecedent to an assertion, looking for an alternative match that produces an alternative, workable binding set. - There are no more antecedents to be matched. In this case, - Use the binding set in hand to instantiate the consequent. - Determine if the instantiated consequent is already asserted. if not, assert it. - Back up to the most recent match with unexplored bindings, looking for an alternative match that produces a workable binding set. - There are no more alternatives matches to be explored at any level.	[Winston, 1992, p. 142]
Search	Children's Search	**Representation**: Semantic Net **Purpose**: Find the children of a specified object u. **Algorithm**: 1) Add children of u's parents to the queue. 2) While the queue is not empty do the following: 2a) Remove a node off the queue (in topological order) and set it equal to v. 2b) If v intersects with children of u and v matches u then remove v's descendants from the queue. 2c) Otherwise v's children to the queue.	[Ellis, 1993, pp. 281-282]
Search	Children's Search	**Representation**: Semantic net **Purpose**: Find the children of a specified object while pruning the search space. **Algorithm**: 1) Add children of u's parents to the queue. 2) Set focus = the intersection of the descendants of u's parents. 3) While the queue is not empty do the following: 3a) Remove a node off the queue (in topological order) and set it equal to v. 3b) If the intersection of the descendants of v's parents with focus is not an empty set then: 3bi) If v in intersection with v and v matches u then set focus = children of v minus the descendants of v. 3bii) Otherwise, Add the children of v to the queue.	[Ellis, 1993, pp. 285-286]

Search	Parent's Search	**Representation**: Semantic net **Purpose**: Find the parents of a specified object u while pruning the search space. Algorithm: 1) Add the root node the queue. 2) Set focus = the intersection of the descendants of the root node's parents. 3) While the queue is not empty do the following: 3a) Remove a node out of the queue in topological order and set it equal to v. 3b) If all of v's parents match and If the intersection of the descendants of v's parents with focus is not an empty set and If v matches u then do the following: 3bi) Mark v as matched. 3bii) Set focus = intersection of the descendants of v's parents with focus. 3biii) Add v's children to the queue.	[Ellis, 1993, p. 282-284]
Search	Topological Search	**Representation**: Semantic net **Purpose**: Find the parents of a specified object u. **Algorithm**: 1) Add the root node the queue. 2) While the queue is not empty do the following: 2a) Remove a node off the queue (in topological order) and set it equal to v. 2b) If all of v parents match and v matches u then 2bi) Mark v as matched. 2bii) Add v's children to the queue.	[Ellis, 1993, p. 281]
Version Space Learning	Candidate Elimination	**Representation**: Version Space Representations **Purpose**: Creates a classification tree for a set of concepts. **Algorithm**: "... *The two components of the version space - the set of maximally general G, satisfactory hypothesis and the set of maximally specific S, satisfactory hypotheses*..." "Initialize by setting G to be the set of maximally general hypotheses and S to be empty. Then, for each presented instance do the following two steps. (1) If the instance is positive then update S so as to ensure that it still contains the set of most specific hypotheses that mutually cover all the seen positives. (2) If it is negative then update G so as to ensure that it still contains the set of most general hypotheses, none of which cover any seen negative. The algorithm terminate if there are no more instances to process or if G is identical to S."	[Thorton, 1992, pp. 29-33]

Version Space Learning	Most General Subsumption	**Knowledge Representation**: Lattice-structured Taxonomy **Purpose**: To operate on a composite description to classify it with respect to an explicit taxonomy. **Algorithm**:" The most general subsumee algorithm also moves downward in waves (like the most specific subsumee algorithm), this time looking for concepts that are subsumed by the input. This algorithm, however is more subtle. Since any concept subsumed by the input will also be subsumed by (all of) its most specific subsumers, the search for subsumees can begin moving downward from the most specific subsumers found by the mss algorithm, without risk of missing any potential subsumees. in fact, the search can safely begin from any one of the mss concepts. When a concept is encountered that is subsumed by the input, it is added to a list of candidate subsumees and is not pursued further. However, very few concepts encountered in the search can be dropped from the wave front in this way. Each concept that is not a subsumee (presumably the normal case) will be replaced in the wave front by all of its recorded specialization's. This is because some further specialization may yet lead to a concept that is subsumed by the input. ... When the mgs search has exhausted all possibilities, then the list of candidate subsumees will become the answer after it is pruned of any candidates that are duplicates or are subsumed by other candidates and hence are not most general. This purging process can, of course, be done incrementally as new candidates are added to the list."	[Woods, 1991, p. 83]
Version Space Learning	Most Specific Subsumption	**Knowledge Representation**: Lattice-structured Taxonomy **Purpose**: To operate on a composite description to classify it with respect to an explicit taxonomy. **Algorithm**: "One version of a most specific algorithm is to start at the top of the lattice and progress downward in waves, maintaining an active list of conceptual descriptions that subsume the input and keeping track of those for who no more specific subsumers are found. ... Note that it is not sufficient to search only below the primary concepts of the input description - there may be concepts that are incomparable with the primary concepts but never less subsume the input."	[Woods, 1991, p. 83]

Appendix L: Representation Table

Class	Name	Representation	Reference
Cognitive Model	Ad Hoc Categories	"Categories that are not conventional or fixed, but rather are made up on the fly for some immediate purpose. Such categories must be constructed on the basis of one's cognitive models of the subject matter under consideration. Examples of such categories are: things to take from one's home during a fire, what to get for a birthday present, what do for entertainment on a weekend... the category is principally determined by goals ..." [Barsalou, 1983, 1984]	[Lackoff, 1987, p. 45]
Cognitive Model	Centrality	The basic members of a category are the most central member. Some members are more central than others therefore better examples of that specific category.	[Lackoff, 1987, p. 95]
Cognitive Model	Chaining	"Complex categories are structured by chaining; central members are linked to other members, which are linked to other member, and so on. For example, women are linked to the sun, which is linked to sunburn, which is linked to the hairy mary grub. It is by virtue of such a chain that the hairy mary grub is in the same category as women."	[Lackoff, 1987, p. 95]
Cognitive Model	Episodic Memory	"Episodic memory, on the other hand, is organized around propositions linked together by their occurrence in the same event or time span. Objects are most commonly defined by their place in a sequence of propositions describing the events associated with an object or an individual. A trip is stored in memory as a sequence of the conceptualizations describing what happened on the trip. Some of the conceptualizations will be marked as salient and some will have been forgotten altogether."	[Schank, Abelson, 1977, p. 18]
Cognitive Model	Experiential Domains	"There are basic domains of experience, which may be culture-specific. These can characterize links in category chains."	[Lackoff, 1987, p. 95]
Cognitive Model	Generators	"Generator categories are defined from central members that are extended by a set of rules. For example, all natural numbers derived by arithmetic from the central numbers (0 - 9).	[Mazlack]
Cognitive Model	Ideal Categories	"Abstract ideal cases which may not be typical or stereotypical. For example, an ideal husband: good provider, faithful, strong, respected, attractive. It provides a structure to set goals as well as to use as a general standard for comparison."	[Mazlack]
Cognitive Model	Image-schematic Model	"Specify schematic images, such a trajectories or long, thin shapes or containers. Our knowledge about baseball pitches includes a trajectory schema."	[Lackoff, 1987, p. 114]
Cognitive Model	Metaphoric Model	It is a model of a mapping of a source image onto a target image where the source and target domains are different from one another.	[Lackoff, 1987, pp. 114, 288]
Cognitive Model	Metonymic Model	Is a model where an individual or subcategory of the category stands for the entire category and this individual can be used as a prototype for the entire category.	[Lackoff, 1987, pp. 114, 203, 288]
Cognitive Model	Prepositional Model	"Specify elements, their properties, and the relations holding among them. Much of our knowledge structure is in the of prepositional models. Thus, a model of a domain would include elements that occur in that domain."	[Lackoff, 1987, p. 113]

Cognitive Model	Radial Categories	Is a cognitive model which has a central cognitive model (a superordinate category) which has arcs radiating from it to other related cognitive models. They can be related by cultural conventions, metaphoric models, metonymic models, etc.. Non central models can be subcenters of other cognitive models.	[Lackoff, 1987, pp. 154, 204, 287] [Mazlack]
Cognitive Model	Salient Examples	"Use a single familiar or memorable case to stand for all in a category. The salient example is used when knowledge is incomplete. For example, judging all vegetarians by a vegetarian friend; thinking that all earthquakes are about the same as the one that you experienced. They are used for making subjective probability judgments with incomplete knowledge."	[Mazlack]
Concept Graph	Concept Graph	Concept graphs contain boxes which represent concepts, circles which represent conceptual relations (derived from Pierce's existential graphs). Concepts represent any entity, action or state that can be described in a language, and conceptual relations show the roles that each entity plays. Conceptual Graphs lend themselves to first order logic with extensions to modal and higher order logics. The concept is divided into two parts by a colon. The first part is called a concept type which labels the concept. The second part is called a referent field with the following values: ∀ - Universal Quantifier. @x - Quantity reference where x is an integer. {*} - Plural referents. {#} - Refers to a contextually known unresolved entity. {xxxx} - Instance, where xxxx is an unique identifier. ? - Question asking which concept ¬ - Negation Concept Graph Example: [Dog: @2] → (Near) → [House]	[Sowa, 1984, pp. 8, 20 -21] [Sowa, 1991, pp. 158 - 172] [Sowa, 1993, pp. 7-10]
Concept Graph	Conceptual Semantics	"Conceptual Semantic structures are built from conceptual constituents that belong to one of a small set of primitive ontological categories, including at least Entity, Place, Path, Action, Event, State, Time, Property, Amount, and maybe according to Jackendoff, also sound, Smell, and Manner. Conceptual constituents can be complex, composed of other conceptual constituents by conceptual functions that map the deeper constituents to the complex constituent." Listed below are some of the primitive categories displayed:" [ENTITY:*x]-(AT)->[PLACE:*y] [PLACE:*x]-(NEAR)->[PLACE:*x] [PLACE:*x]-(ON)->[SURFACE:*y] [PLACE:*x]-(IN)->[VOLUME:*y] [EVENT:*x]-(theme)->[ENTITY], -(along)->[PATH] [PATH:*x]-(TO)->[PLACE:*y] [PATH:*x]-(FROM)->[PLACE:*y] and etc. ,...."	[Willems, 1993, pp. 314-325]

Concept Graph	Well-Formed Conceptual Graph	"At a very elementary level, we define a conceptual graph relative to a support, which provides background knowledge on a specific domain application. In a support, we group: - T_c, a set of concept types, structured in a lattice, with T as supremem (the universal type) and $\perp$ as infimum (the absurd type), - T_r, a set of relation types, which is not structured - but could be partially ordered. - a set of markers for concept vertices: one generic marker * and a set M of individual markers which allow naming of distinct entities: $m \cup \{*\}$ is provided with the order, such that * is greater than all markers, and any two individual makers are non-comparable. - a signature for each relation type, that we represent with a star graph; each star graph fixes the arity of a relation type and shows the greatest concept types this relation type can link. The set of all start graphs is called the basis of the support, and denoted by β_0. A conceptual graph G = (R,C,U, lab) is a bipartite, connected, finite graph. R and C denote the two classes of relations and concept vertices, which we also call r-vertices and c-vertices. U is the set of edges, and the edges adjacent to each r-vertex are totally ordered. Every vertex has a label defined by the mapping lab. If $c \in C$, then $lab(c) \in T_c \times M \cup \{*\}$; if $r \in R$, then $lab(r) \in T_r$. G is said to be well-formed relative to a support s if it conforms to the constraints defined by the star graph set of S. Any start graph of β_0 can be identified with the naturally associated conceptual graph: B_{tr}, the star graph of the relation type tr, is a well formed conceptual graph, having a unique r-vertex labeled tr, whose neighbors are generic c-vertices."	[Mugnier, Chein, 1993, pp. 295-296]

Conceptual Dependency	Conceptual Dependency	1) Physical object in the world can perform actions, represented by object <=> action. Physical objects are called Picture Producers (PP)s and actions are called ACTs. The previous representation becomes PP <=> ACT. 2) ACTs can perform actions; have past, present and future tenses; and have the following conceptual relations: *Objective* - object performed upon by an ACT. Directive - ACTs can have direction indicating old and new locations of an object. *Recipient* - object involved in a transition and has to and from arrows. *Instrumental* - ACTs can have "instruments" which are conceptualizations on how an ACT is performed. 3) There are 11 primitive ACTs in which all verbs are broken down into, they are divided into 4 groups: *Physical Actions* PROPEL - apply force to an object MOVE - physical movement of a body part. INGEST - To take something into the body. EXPEL - To expel something from the body GRASP - Grasp an object. *Transitions* PTRANS - Change location of a physical object. ATRANS - Abstract transfer of an object. *Communication* SPEAK - Create Sounds. ATTEND - Sense a stimuli. *Mental actions* MTRANS - Transfer thoughts or information mentally. MBUILD - Create and manipulate thoughts. 4) ACTs can cause consequences, which are represented by 3 line horizontal arrow that points to the causee. There are 3 types of causation that are explained below: *Reason* - The action is only the cause for another action. *Result* - When an action causes an object to become a particular state or state change. *Enabling* - When an action causes a state change which enables another action to take place. 5) Other conceptual categories: LOC - Location where a physical ACT takes place. T - Conceptualization time. AA - Action modification features PA - Object attribute written in the form of STATEs. 6) STATEs describe a sate of an object and is represented numerically. Some examples of states are HEALTH, FEAR, ANGER, MENTAL, PHYSICAL, etc. 7) Conceptual dependency deals with trying to find context of the text as it was written and what inferences the speakers were making. 8) The mind is made up of 3 parts: conscious processor (processes memory), intermediate memory (holds expressive, and immediate memory) and long term memory (hold information for recall).	[Brachman, 1979, p. 15] [Harris, 1985, pp. 255-263] [Schank, Colby, 1973, pp. 187-248] [Schank, 1975, pp. 22-82], [Schank, Reisbeck, 1981, pp. 10-26]

Decision Tree	Decision Tree	Decision tree is a hierarchy of nodes where the leaf nodes contain class names and the non-leaf nodes are procedures which test an object's attributes to indicate which branch an object is to follow until it reaches its classifying leaf node. The classifying leaf nodes contain information about objects classified in their domain.	[Richman, 1991, p. 104] [Utgoff, 1989, pp. 162, 165]
ER Model	Entity-Relationship Model	The ER model has two primitives called entities and relations. An entity has name, a set of attributes, and a key which is a non-empty subset of the previously mentioned attributes that uniquely identifies the entity. A relation also has a name and a key made up of non-empty subset of attributes that uniquely identifies the relation. Attributes can have one or more values. Entities and relations can have one or more keys. Recurrent relations and relationships based on dependent identifiers are not allowed. Example is listed below: *relation* - Lecture = (Professor, Course, Room, Qtr) *entity* - Professor(1234, Name, Degrees) *entity* - Student(SSN, Name, Major) *combination* - Lecture(1234, Comp-101, 4a, Fall94)	[Creasy, Ellis, 1993, pp. 127-128] [Ling, Lee, 1992, p. 264] [Thalheim, 1992, p. 9]
ER Model	ER+ Model	The ER+ model is similar to the ER model with the following additional features. 1) A relationship type that connects an entity type to an instance and who's mapping is constrained by its predicate cardinality. 2) An entity type indicates the domain an entity lies and must have least one attribute associated with it. 3) Entity and relationship types have a time attribute the last time they modified or when they were created. 4) The is_subset_of and is_covered_by relationships were added.	[Song, Johannesson, Bubenko, 1992, pp. 101-102]
Feature Bundle	Classical Feature Bundle View	"A feature bundle is a collection of properties. The elements in the ontology are properties. Structurally, the bundle is characterized by a CONTAINER schema, where the properties are inside the container. Classical categories can be represented by feature bundles."	[Lackoff, 1987, p. 286]
Feature Bundle	Probabilistic Feature Bundle View	"As usual, the first assumption is that the representation of a concept is a summary description that applies to all of its instances. the second assumption has two parts: (1) any dimension used to represent a concept must be a salient one, some of whose values have a substantial probability of occurring in instances of the concept; and (2) the value of a dimension represented in a concept is the (subjective) average of the values of the concept's subsets or instances o this dimension."	[Smith, Medin, 1981, p. 102]
Feature Bundle	Weighted Feature Bundle	"A feature is a symbol representing a property. A feature bundle is an unstructured set of features, representing a set of properties. A weighted feature bundle assigns weights to the features in a bundle; the weights are used to account for prototype effects in the prototypical category member. Approximations to the prototype are defined in terms of shared features. Deviation from the prototype in highly weighted features places a member further away from the prototype than deviation in a less highly weighted feature."	[Lackoff, 1987, p. 115]

Frames	Frame System	A frame system represents knowledge in prefabricated template like structures, called frames, using stereotype and prototype models. Frames can represent situations, concepts, actions and individuals. Each frame has one or more slot and value pairs that represent concept attributes and their corresponding values. Default values are usually supplied to the frame slots, when the frame is formed, unless they are otherwise overwritten. "AKO" slots give the system a hierarchical structure and "ISA" slots represent instances. Frame slots can point to other sub-frames; procedures on: how the frame is to be used, exception processing and frame transactions; superordinate frames; and subordinate frames. Slots can be added or deleted from frames during the life of the frame system.	[Brachman, 1991, pp. 403-405] [Crawford, Kuipers, 1991, pp. 301-303] [Galloway, 1984] [Kuipers, 1975, p. 152] [Minsky, 1975, p. 212] [Sowa, 1984, pp. 19-20] [Winston, 1992, p. 200]
Fuzzy Sets	Fuzzy Set Theory	Fuzzy set theory is a logic about a collection of entities contained in the universe U, where set membership is measured on a scale between 0 and 1 inclusive. A membership degree of 0 indicates an entity does not belong to the set. A membership degree of 1 indicates that a entity belongs to the set. A membership degree of .5 indicates that it is ambiguous as whether the entity belongs to the set or not. A membership degree of .9 indicates an entity is very likely to belong to a set and a degree of .1 indicates an entity is very unlikely to belong to a set. Some operations that can be performed on fuzzy sets are: union, intersection, not, concentrate, dilate, normalize, intensify and fuzzify.	[Lackoff, 1987, pp. 287-288] [Schmucker, 1984, pp. 5-18] [Smith, Medin, 1981, p. 180] [Sowa, 1991, p. 5]
Hierarchy	Class Network	Class network is a hierarchical network of classes. "Each class has an intention, a current extension, and a set of practical conditionals. The intention defines a predicate for class membership and takes the form of a conjunction of observable features of objects. The intention need not contain terms that appear in the intentions of the class's superclasses, since these will be inherited. In certain cases this means that there are no new terms in the intention of a class: such classes are given the intention term TRUE, which is satisfied by any object. Disjunctive concepts are represented by forming separate classes for each disjunction term. The extension is a list of all currently observable objects that have satisfied the intention."	[Scott & Markovitch, 1991, pp. 394-395]
Hierarchy	Classical Hierarchy	The classical hierarchy has the following features: (1) Categories at the same level don't overlap. (2) Categories inherit all the attributes of their ancestors. (3) For a entity to belong to a category its attributes must agree with its classifying category. (4) A category is a group of entities that have one or more attributes in common. (5) An entity is defined by a superordinate category and a set of attribute values that differentiate it from the other entities in its category. (6) An entity may have only one parent. (7) Started by Aristotle. (8) It is acyclic. (9) Each class may inherit at most one immediate superclass	[Lackoff, 1987, pp. 161,287] [Sowa, 1984, p. 16] [Smith, Medin, 1981, pp. 23-25] [Touretzky, 1986, pp. 2,70-71,76] [Woods, 1991, pp. 69-70]

Hierarchy	Multiple Inheritance Hierarchy	" We derive tokens from real-world individuals and predicates by prefixing them with a sign: one of +, -, or #. Tokens are distinguished by signs as positive (+), negative (-), or neutral (#), respectively; they may also be distinguished by type as either predicate or individual tokens. ... The symbol Π shall denote the set of all real-world individuals and predicates referred to in an inheritance system. The system Θ shall denote the set of tokens derivable from the objects in Π and the three signs. Formally, we write: $\Theta = \{+,-,\#\} \times \Pi$. Inheritance assertions are elements of Θ x Θ, i.e. they are ordered pairs of tokens. There are six types of well-formed ordered pairs; three of these contain an individual token as the first element; the other three contain a predicate token. <+a,+p> - a is a p. <+a, -p> - a is not a p. <+a, #p> - No conclusion whether a is a p. <+p, +q> - p's are q's. <+p, -q> - p's are not q's. <+p, #q> - No conclusion whether p's are q's. ... Note that the first element of an inheritance assertion may be either an individual or a predicate but is always positive, while the second element must be a predicate but may be either positive, negative, or neutral. ."	[Touretzky, 1986, pp. 31-33]
Hierarchy	Path Based Inheritance System	A path based inheritance system consists of finite number of nodes interconnected by arcs called edges. Edges can have a value of 0 or 1, a 0 representing a negative relationship and a 1 representing a positive relationship, this is also referred to as the edge's polarity. The inheritance system is acyclic. A path is a sequence of edges. Its polarity is determined by the last edge's polarity. Two paths can contradict if their polarity are opposites. A path is preempted if it has an intermediate edge with a polarity that is the opposite of the path's polarity. Paths are concatenated if one path takes up where the other path leaves off. A path α is inheritable in β if α is a concatenation of paths in β, α and β don't contradict, and α is not preempted in β. β is a credulous grounded extension of a set of edges Γ (a set of edges connecting two nodes) iff $\Gamma \subseteq \beta$ and for all α, $\alpha \in \beta$-Γ iff α is inheritable in β.	[Selman, Levesque, 1991, pp. 285-290] [Stein, 1991, pp. 271-272]
Hierarchy	Structured Inheritance Network (SI-Net)	"A structured inheritance network has a fixed set of node and link types (the number of which is small), thereby providing the basis for a fixed and well-defined interpreter. The links in this kind of net are used to explicitly express 'epistemological' relationships between Concepts and their parts ("Roles" and "Structural Descriptions"), that is, structuring relationships between formal objects used to represent knowledge."	[Brachman, 1979, pp. 4-5]

Hierarchy Multiple Inheritance	Hierarchical Knowledge Base (HKB)	"The class hierarchy is a directed, acyclic, graph over the predicates of the language. A node may have more than one parent, but there is no notion of inheritance cancellation. The presence of a link from a node P to a parent node Q expresses, as usual, the logical implication $\forall x\ P(x) \supset Q(x)$. For now, we assume that all non-primitive classes (i.e., non-leaf predicates) in the hierarchy are equivalent to the union of the classes they subsume (e.g., Mammal is exhausted by its subconcepts, Dog, Cat, etc.). This comprises a form of intentional closure, corresponding to Clark's [1978] 'Predicate Completion'. Logically, this amount to the assumption that non-primitive predicates are equivalent to the disjunction of their immediate children."	[Borgida, Etherington, 1989, p. 34]
Higraph	Higraph	"A higraph is a quadruple $H = (B,\sigma,\pi,E)$ where B is a finite set of blobs and E is a set of edges (a binary relation on blobs). The subblob function $\sigma: B \rightarrow 2^B$ assigns to each blob $x \in B$ the set $\sigma(x)$ of its subblobs; it is assumed that the function is cycle free. The function $\pi: B \rightarrow 2^{B \times B}$ introduces an equivalence relation on the blobs; these classes of equivalence for a given blob x will be denoted as $\pi i(x)$. The atomic blobs, denoted by A, are those that do not have any subblobs."	[Zadrozny, 1991, p. 367]
Lattice	Lattice	A lattice is a representational network with conjunctive and disjunctive operators. The operators satisfy the following identities {MackLane, Birkhoff, 1967]: 1. $x \wedge x = x \vee x = x$ (Idempotent) 2. $x \wedge y = y \wedge x =, x \vee y = y \vee x$ (Commutative) 3. $x \wedge (y \wedge z) = (x \wedge y) \wedge z$ (Associative) 4. $x \vee (y \vee z) = (z \vee y) \vee z$ (Associative) 5. $x \wedge (x \vee y) = x \vee (x \wedge y) = x$ (Absorption) 6. $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ (Distributive) 7. $x \vee (y \wedge x) = (x \vee y) \wedge (x \wedge z)$ (Distributive) 8. $x \wedge \sim x = 0, x \vee \sim x = 1$ 9. $\sim(\sim x) = x$ 10. $\sim(x \wedge y) = \sim x \vee \sim y$ 11. $\sim(x \vee y) = \sim x \wedge \sim y$	[Touretzky, 1986, p. 79-80] [Woods, 1991, p. 81]
Logic	Default Logic	Default logic is a first order logic extended with default rules and defined by the 3-tuple $<\alpha,L,\omega>$, where: α - Is the default prerequisite written as a wff. ω - Is the inference written as wff. L - Is the justifications written as wff. Default rules are written in the form of : where $L = \{\beta_1, ..., \beta_n\}$which follows the logic if α holds and if the β's can be assumed then conclude ω.	[Marek, Truszcynski, 1989, p. 278] [Rajasekar, Lobo, Minker, 1989, p. 350]
Logic	Intentional Logic	"Intentional logic's are closely related to modal logic. They are used to represent prepositional attitudes, or they are used to represent verbs that express some mental attitude toward a proposition. Such verbs include know, believe, think, hope wish, fear, and imagine. If the only intentional verbs are know and believe, the logic is called epistemic logic."	[Sowa, 1991, p. 5-6]

Logic	Minimal Modal Logic K	" (pL) all prepositional tautologies are axioms of K (K) $\vdash_K \mathrm{o}p(p \rightarrow q) \rightarrow (\mathrm{o}p \rightarrow \mathrm{o}q)$, (N) if $\vdash_K p$, then $\vdash_K \mathrm{o}p$ (Necessitation), (MP) if $\vdash_K p$ and $\vdash_K p \rightarrow q$, then $\vdash_K q$ (Modus Ponens). The system K can be used as a basis for deontic logic, in which case o is interpreted as 'desirability', or as describing believe, in which case o is interpreted as 'believes that'."	[Berg, 1993, pp. 414-415]
Logic	Modal Logic	Modal logic expresses the English modal auxiliaries: can, may, will, and must in the form of ◊ to represent possibility (may, can will) and o to represent necessity (must). The formula op states that p is necessarily true, and the formula ◊p states that p is possible. Possibility is defined in the terms of necessity which gives o: ◊p = ~o~p.	[Berg, 1993, p. 414] [Sowa, 1991, p. 5]
Logic	Modal Logic S4	"(pL) all prepositional tautologies are axioms of S4: (K) $\vdash_{S4} \mathrm{o}p(p \rightarrow q) \rightarrow (\mathrm{o}p \rightarrow \mathrm{o}q)$, (T) $\vdash_{S4} \mathrm{o}p \rightarrow p$, (4) $\vdash_{S4} \mathrm{o}p \rightarrow \mathrm{oo}p$, (N) if $\vdash_{S4} p$, then $\vdash_{S4} \mathrm{o}p$ (Necessitation), (MP) if $\vdash_{S4} p$ and $\vdash_{S4} p \rightarrow q$, then $\vdash_{S4} q$ (Modus Ponens)."	[Berg, 1993, p. 415]
Logic	Modal Logic S5	"(pL) all prepositional tautologies are axioms of S5: (K) $\vdash_{S5} \mathrm{o}p(p \rightarrow q) \rightarrow (\mathrm{o}p \rightarrow \mathrm{o}q)$, (T) $\vdash_{S5} \mathrm{o}p \rightarrow p$, (4) $\vdash_{S5} \mathrm{o}p \rightarrow \mathrm{oo}p$, (5) $\vdash_{S5} ◊p \rightarrow \mathrm{o}◊p$, (N) if $\vdash_{S5} p$, then $\vdash_{S5} \mathrm{o}p$ (Necessitation), (MP) if $\vdash_{S5} p$ and $\vdash_{S5} p \rightarrow q$, then $\vdash_{S5} q$ (Modus Ponens). The system S5 is often seen as axiomatization of knowledge."	[Berg, 1993, p. 415]
Logic	Modal Logic T	"(pL) all prepositional tautologies are axioms of T: (K) $\vdash_T \mathrm{o}p(p \rightarrow q) \rightarrow (\mathrm{o}p \rightarrow \mathrm{o}q)$, (T) $\vdash_T \mathrm{o}p \rightarrow p$, (N) if $\vdash_T p$, then $\vdash_T \mathrm{o}p$ (Necessitation), (MP) if $\vdash_T p$ and $\vdash_T p \rightarrow q$, then $\vdash_T q$ (Modus Ponens). The system T (which is sometimes called M) can be used to describe knowledge, in which case o is interpreted as 'knows that '. Another application of T is that it can be used for the specification of user interest in the context of data retrieval."	[Berg, 1993, p. 415]
Logic	Monotonic Logic	"Monotonic logic is standard logic. It is called monotonic because the number of provable theorems increases monotonically as the number of assumptions increases. Adding a new axiom can never cause a previous theorem to become unprovable. If the new axiom causes a contradiction, then everything becomes provable."	[Sowa, 1991, p. 4-5]

Logic	Non-Monotonic Logic	"Non-monotonic logic is the name for a family of new logic's used to represent defaults and exceptions. Tweety the penguin is a commonly used example. If tweety is a bird, then one might assume that Tweety can fly. But the additional information that Tweety is a penguin should block the proof that Tweety can fly. That kind of blocking, which is characteristic of non-monotonic logic's, is not possible in standard logic's."	[Sowa, 1991, p. 5]
Logic	Non-Monotonic Modal Logic	"The language ι of the non-monotonic logic consists of the first order logic language augmented with the modal operator M, and allows sentences such as Mp, ¬Mp where p is a first order formulate. The logic uses the monotonic inference rules of Modus Ponens and generalization along with a new rule of the form: *If* $\forall_T \neg p$, *then* $\vdash_T Mp$ which means that if the negation of a formula is not derivable from T then it may be consistently inferred by the formula."	[Rajasekar, Lobo, Minker, 1989, p. 350]
Logic	Sorted Logic	"Sorted logic restricts each variable to a specific sort. In standard logic, a quantifier like (∀x) is completely unrestricted, and x could range over any entity in the universe. In sorted logic, however, a quantifier like (∀x:DONKEY) limits x to entities of the sort DONKEY. The sorts of sorted logic correspond to the types of a semantic network, and the same kinds of inheritance mechanisms may be used improve the efficiency of proof procedures."	[Sowa, 1991, p. 5]
Logic	Temporal Logic	"Temporal logic's deal with time, which raises complication that are not handled by the static models of standard logic. Some versions of temporal logic have a close affinity with modal logic, with a small box symbol representing always, and the small diamond symbol representing sometimes. Tense logic's represent the multiple reference times implied by the tenses in natural languages."	[Sowa, 1991, p. 6]
Probabilistic Model	Probabilistic	"A concept is defined by a collection of features and everything that has a preponderance of those features is an instance of that concept. ... It is also the basis for modern techniques of cluster analysis."	[Smith, Medin, 1981, pp. 102-109]
Probabilistic Model	Probabilistic Concepts	"... Probabilistic concepts, which associate a probability or weight with properties of a concept definition. For example, an independent cue model stores the conditional probability, $P(f\|C_k)$, or other weight of each feature f's presence with respect to each category C_k. Classification involve summing the weights of features that are present in a new observation until the sum passes a specified threshold . More typical instances will tend to have features with higher weights, and typicality phenomena are explained by the differences in the time required to reach some criteria sum. ... limited to recognizing linearly separable categories."	[Fisher, Pazzani, 1991, p. 6]

Production System	Production System	"A production system consists of a production memory and a working memory. Working memory, WM, is a collection of memory elements. Production memory, PM, is a collection of production; each of which is a conditional statement composed of zero or more condition elements and zero or more action elements (written P: (C1 C2 . . . A1 A2 ...)). Whenever each condition element in a production is matched by a memory element in WM, the production is satisfied and may be fired. When a production is fired, each action element in the production is executed. This ordinarily causes WM to be modified in some way; for example a new element might be added to WM or an element in WM might be deleted. The basic unit of behavior is the recognize-act cycle. The cycle has two parts: (1) discovering the conflict set, the subset of satisfied productions, and selecting one or more of the productions in the conflict set as the productions to be fired; and then (2) firing the productions selected, that is executing their action elements. Processing continues, cycle follows cycles, until either no production is satisfied or an action element explicitly stops the processing."	[McDermott et al, 1978, p. 156]
Prototype	Prototype	A prototype is an "instance" (concept) of a class that represents the typical concept for that class. It is used to measure class membership of other concepts within that class or whether to classify a non-member concept within that class. The prototype is the central member of its category and its features are the typical features of its category.	[Mazlack] [Smith & Medin, 1981 in Sowa, 1984, p. 17] [Sowa, 1984, pp. 136 - 137]
Schematic	Schema	Schema is a knowledge representation that mimics local (domain-specific) incomplete background knowledge humans use for contextual inferencing in interpretation, planning, and concept construction. It is information gained from past experience represented by clusters of entities, attributes and events (called a schematic cluster). Formal definition of schematic cluster is given below: "A schematic cluster for a type t is a set of monadic abstraction $\{\lambda a_1 u_1, \ldots, \lambda a_n u_n$" where each formal parameter is a of type t. Each abstraction $\lambda a_i u_i$ in the set is called a schema for the type t."	[Sowa, 1984, pp. 128-137]
Schematic	Simple Proposition	"A simple proposition consists of an ontology of elements ("the arguments") and a basic predicate that holds of those arguments. the overall structure of the proposition is thus characterized by a part-whole schema, where the proposition = the whole, the predicate = a part, and the arguments = the other parts. In addition, certain semantic relations may hold among the arguments: there may be an agent, a patient, an experience, an instrument, a location, etc. Semantic relations are represented structurally by link schemas, and the kinds of schemas are represented by assignments o links to categories of relations."	[Lackoff, 1987, p. 285]

Semantic Net	Extended Semantic Net	Extended semantic networks can encode traditional logic, represent time, represent quantification and can impose topic hierarchies upon the net. The nodes in the system can have n-ary relationships in which class, individual, adjective, time, location arguments can be expressed. The relationships can either be logical connectors, modal operators, or verbs. The representations when worded out take the form of predicate calculus where the predicate is placed after the first argument. Every predicate includes a time range. The topic hierarchy is mapped using specialization and generalization predicates. Subconcepts referenced using the specialization predicate and the super concepts referenced using the generalization predicate.	[Harris, 1985, pp. 243-249]
Semantic Net	Human Associative Memory (HAM)	"The elements of HAM were propositions, binary trees that represented the underlying structure of sentences.... Relations allowed between nodes in the trees included set membership and subset, some cases like subject, object, location, and time, and some logical indicators like predicated, context, and fact - all represented uniformly. Propositions in HAM had truth values, and were supposed to convey assertions about the world."	[Brachman, 1979, p. 17]
Semantic Net	Prepositional Semantic Net	"A prepositional semantic net - is a semantic network in which every proposition represented in the network is represented by a node rather than by an arc. isolated nodes are not allowed in the network."	[Shapiro, 1991, p. 137]
Semantic Net	Semantic Net	A semantic net is a knowledge representational model that has the following characteristics: 1) Nodes are connected by labeled directed arcs that represent binary relations. 2) It is acyclic. 3) It usually has a class hierarchy. 4) Nodes inherit properties from their superordinate classes. Typically "ISA" relations are used to defined semantic net hierarchies. 5) Nodes can represent physical objects, situations, events, actions, sets, attributes, etc.. 6) Nodes also can represent classes of objects, objects, or instances of an object. 7) Every concept is represented by an unique node. 8) Its purpose is to represent concepts expressed in natural language sentence and phrases. 9) There are no unconnected nodes. 10) A node can have 0 to n arcs emanating from it.	[Biebow, Chaty, 1993, p. 76] [Brachman, 1979, pp. 5-6, 27, 34-36] [Harris, 1985, pp. 219-220] [Hendrix,1978, p. 126] [Hendrix, 1979, p. 53] [Rumelhart, 1972, pp. 201-209] [Shapiro, 1991, pp. 137, 179] [Simmons, 1973, p. 63] [Sowa, 1984, p. 76] [Winston, 1992, p. 20]

Semantic Net	Semantic Net Procedural	In a procedural semantic network there are three aspects to representation: classes, relations, and procedures (which set procedural semantics apart from regular semantic nets). Instances with common features are grouped into classes. The four basic operations that can be performed upon classes are instance: creation, deletion, retrieval and membership. Classes are referred to as primary nodes while instances are called secondary nodes. The directed labeled arcs connecting classes together are called relations and are treated like nodes. They map classes from one domain to another. Some operations performed on relations are testing if a relationship exists or not exist between two objects. Procedures are considered a class whose instances are called processes and correspond to program activation's. The sense of word is defined by a set of processes and these processes determine if a selected word should be applied to an object.	[Clark, Clark, 1977, p. 439] [Levesque, 1979, pp. 95-99] [Norman, Rumelhart, 1975, pp. 35-40]
Semantic Net	Semantic Net Partitioned	A partitioned semantic network is similar to a semantic network except that the nodes and arcs are divided up into spaces. A space is group of related nodes and arcs which are cross indexed and can be manipulated like individual nodes and arcs. Nodes and arcs can reside in one or more spaces. A group of spaces bundled together is called a vista. Partitioned semantic net operations usually take place at the vista level and work only with the specified vista unless other information is needed from another vista. This is very efficient since the entire does not need to be traversed. A super node is created with a group of vistas and/or spaces bundled together. This is for complex events and stories.	[Harris, 1985, p. 230] [Hendrix, 1978, pp. 132 - 137] [Hendrix, 1979, pp. 59-64]
Semantic Net	Semantic Net with Cycles	A semantic net with cycles is similar to the semantic net above except it allows cycles in its network.	[McDonald, Hayes-Roth, 1978, pp. 433-434]
Semantic Net	Semantic Net "SNePS"	SNePS should be able to portray anything expressed in a natural language. The network consists of nodes and labeled directed arcs. There are four different types of nodes: base, variable, molecular, and pattern. Base nodes have node arcs leading from them and represent unique entities that are different from anything else represented in the network. Variable nodes also have no arcs emanating from them and represent individuals and propositions. Molecular nodes have arcs that emanate from them and represent propositions and rules. Pattern nodes send out arcs and represent open sentences translated in predicate logic. All nodes have their own identifiers. SNePS is capable of learning by being explicitly told information, by deduction, and induction from examples. Arcs are labeled by the user and are written terms of grammatical punctuation to express predicate calculus.	[Shapiro, Rapaport, 1992, pp. 243-275]

Semantic Net	Semantic Net S-Net - A bare bones semantic net	"To define s-nets, we begin with vertices, defines as the following type of expressions: *constant vertices* - c1, c2, ... *variable vertices* - x1, x2, ... *function vertices* - $f1^1$,$f1^2$, ..., $f2^1$,$f2^2$, ... (upper index indicates audacity) *predicate vertices* - $P1^1$,$P1^2$, ... $P2^1$, $P2^2$, ... *quantifier vertices* ∀, ∃ *operator vertices* - ∧, ∨, ⊃, ~ *term vertex*: any constant or variable vertex, or any expression of the form $(f_i^n\ t_1, ...t_n)$ where t_1, ..., t_n are term vertices. *Prepositional vertex*: any expression of form $(P_i^n\ t_1...t_n)$ are term vertices; or any expression of form (Q x R) where Q is a quantifier vertex, x is a variable vertex, and R is a prepositional vertex; or any expression of form (∨ R S), (∧ R S), (~ R), or (⊃ R), where R and S are prepositional vertices. We use labeled-edge terminology to describe the relation between an expression and its immediate (ordered) constituents. In a vertex V of form $(O\ V_1...V_n)$ (where necessarily) is a function, predicate, quantifier, or operator vertex), we say that there is a labeled detected edge (V, O, OP) from vertex V to vertex O and a labeled directed edge $(V, V_i,$ i) from vertex V to vertex V_i for i = 1, ..., n. Finally we define an s-net as a finite set of prepositional vertices."	[Schubert, 1991, p. 99]
Semantic Net	Three-Level Semantic Net	"1. Reasoning takes place in a three-level structure consisting of a reverential level R (a partial order of theories representing background knowledge), an object level (describing current situation), and a metalevel (constraining types of models). 2. The referential level is a collection of graphs whose nodes correspond to theories describing background knowledge, and who's links represent the partial orderings on these theories. 3. An object level theory T is augmented by theories from the referential level by extending paths from the subformulas of T to corresponding theories of R. The resulting new, consistent, theories PT(T) can be then further extended to PT(PT(T)), and so on. 4. Models of such extensions are then built, subject to metalevel constraints."	[Zadrozny, 1991, pp. 372-373]

Appendix M: Conceptualization Problems

Name	Problem	Reference
Ambiguity	Representation: Inheritance Systems "Shortest path reasoners fail to recognize ambiguity."	[Touretzky, 1986, p. 212]
Ambiguity	"Ambiguity ought to be the bane of comprehension because many - probably most - sentences have more than one interpretation, or reading. Although people ought to have great trouble selecting the intended reading, in practice they are rarely aware of more than one reading, which they select immediately."	[Clark, Clark, 1977, p. 80]
Analogies	Interpreting analogies like: foot of a mountain; foot of a list; mapping or projecting the human body onto other things.	[Lackoff, 1987, pp. 19 - 20]
Atypically	"Present day inheritance systems provide no way to assert that an individual is atypical nor can they determine the relevance of one atypical property to some other property."	[Touretzky, 1986, p. 27]
Changing definitions and standards	Dealing with the problem of changing definitions and standards, such as stock market terms.	[Sowa, 1984, p. 296]
Comparative Terms	"How to evaluate comparative terms like good, poor, big, and small. i.e. To answer the following questions a computer must determine whether big refers to height, area, volume or number of students: How big is the tree? How big is the form? How big is the pot? How big is the university?"	[Sowa, 1984, p. 295]
Compounds	"Many examples, i.e. topless bar, electrical engineer, ect., are what linguists call "compounds". It is often the case that the meanings of compounds are not compositional; that is the meaning of the whole cannot be predicted from the meanings of the parts and the way they are put together. The parts do play a role in the meaning of the whole expression - they motivate that meaning- but more is required, a relevant ICM where each part of the compound fits some element of the ICM."	[Lackoff, 1987, p. 147]
Conceptual Abstraction	"Representing the same concept with two different description or the same concept has multiple descriptions which can be used independently to describe the concept. i.e. polygon with 3 sides is equivalent to a polygon with 3 angles, equivalent to a triangle."	[Woods, 1991, pp. 52-53]
Conflicting Categorization	"The question of conflicting categorization strategies is real. For example, [Gould, 1980] in 'What, If Anything Is A Zebra?" discusses the sometimes opposing categorization strategies of two different schools of biologists: pheneticists and cladists. The cladists are concerned with the historic, evolutionary branching order (i.e., the production of genealogical 'family trees'). In contrast, the pheneticist look at overall similarity in form, function, and biological role."	[Mazlack]

Conflicting Categorization	Dixon summarizes how different groups of people have classified nouns: **Ancient Greeks** Determiner : Orthographic endings. Classes : masculine, feminine, and inanimate **Indo-Europeans** Determiner : Semantic categories of animateness and sex. Classes : masculine, feminine, and inanimate **Romance Languages** Determiner : Corresponding Latin noun classification Classes : masculine and feminine **Tshukwe group of Khosan languages** masculine - males, strong, and tall or slender objects. feminine - females, weak, and short or round objects. common - includes everything else **Bhatnu** - Classification has no correlation with gender types. **Swahili** class 1 - inanimate things class 2 - non-human living things class 3 - name of human beings class 4 - collective nouns **Dyirbal** Bayi - animateness, human masculinity Balan - human femininity, water, fire, fighting Balam - non-flesh food Bala - everything else **Yidiny** - Not all the nouns are classified and some may be classified in more than one category which can be decided by syntax or grammar. Bama - with any noun referring to human beings Minya - non-human animates Mayi - non-flesh food Bana - nouns referring to liquids Buri - nouns dealing with fire Jugi - types of wood and wood things. Wirra - everything else.	[Dixon, 1982, pp. 159 - 183]

Conflicting Categorization	"**Numerical phenetics** - ...classification be based exclusively on 'overall similarity.' They also proposed, in order to make the method more objective, that every character be given equal weight, even though this would require the use of large numbers of characters (preferably well over a hundred). In order to reduce the values of so many character to a single measure of 'overall similarity', each character is to be recorded in numerical form. Finally, the clustering of species and their taxonomic distance from each other is to be calculated by the use of algorithms that operationally manipulate characters in certain ways, usually with the help of computers. The resulting diagram of relationship is called a phenogram. The calculated distances can be converted directly into a classification. **Cladistics** - ... bases classifications exclusively on genealogy, that is, on the branching pattern of phylogeny. For the cladist phylogeny consists of a sequence of dichotomies, each representing the splitting of a parental species into two daughter species; the ancestral species ceases to exist at the time of the dichotomy; sister groups must be given the same categorical rank; and the ancestral species together with all of his descendants must be included in a single 'holophyletic taxon. **Evolutionary classification** - ... 'evolutionary' school, which bases its classification on observed similarities and differences among groups of organisms, evaluated in the light of their inferred evolutionary history. the evolutionary school includes in the analysis all available attributes of the organisms, their correlation's, ecological stations, and patterns of distributions and attempts to reflect both of the major evolutionary processes, branching and the subsequent diverging of the branches."	[Mayr, 1984, pp. 646-647]
Context	"Communication through a natural language is, in large part, a function of context. Where and when something is said largely determines what is said."	[Odell, 1981] in [Sowa, 1984,p. 341]
Contextual Inferences	"In understanding language, people make predictions all the time about the sentences and the pieces of each sentence they hear. The predictions may be wrong at times, but they are an important part of the understanding process. (Suppose we had a sentence: 'Fred likes vanilla milkshakes', followed by 'John likes chocolate,' then the sentence doesn't mean that 'John likes eating chocolate', but that 'John likes drinking chocolate milkshakes'.	[Schank, 1975, pp. 16-17]
Continuity	"the concept expressed by any given word in a natural language is inextricably tied to the concepts expressed by nearly every other word in the language. While the words themselves are no doubt discrete, the concepts they involve, or are tied to, are continuous with other concepts."	[Odell, 1981] in [Sowa, 1984,p. 342]
Emphasis	"What we mean is also a function of how we say it. Where or upon what word or words we place an emphasis (intonation contour) as well as how we move various parts of our bodies (body language) will frequently affect what we mean."	[Odell, 1981] in [Sowa, 1984,p. 341]
Family Resemblance	"Wittgenstein (1953) showed that ordinary words like game have no common properties that characterize all their uses. Competition is present in ball games, but absent in solitaire or ring around the e rosy. Organized sport follows strict rules, but not spontaneous play. And serious games of life or war lack the aspects of leisure and enjoyment. The concept GAME has no differentiate that distinguish games from all other activities. Instead games share a sort of family resemblance:... Wittgenstein maintained that most words are define by family resemblances. He considered the meaning of a word to be its use within the language... "	[Sowa, 1984, p. 15]

Family Resemblance	"What most, if not all general empirical terms mean in a natural language, as opposed to what we might mean by them on some specific occasion, cannot be specified formally, that is in terms of necessary and sufficient conditions. They are family resemblant in nature."	[Odell, 1981] in [Sowa, 1984,p. 342]
Family Resemblances	Categorizing objects in a class that don't have anything in common, i.e. games don't always have things in common but are still categorized under the heading "games".	[Lackoff, 1987, p. 16]
Frame Conflict	Frame conflict occurs when there is a conflict in how things should be clustered, what should be the relevant hierarchical level, and what is the category subordination,	[Mazlack]
Hedges	Examples of Hedges are: strictly speaking, loosely speaking, and technically speaking.	[Lackoff, 1987, p. 138]
Homonym	A word with more than one meaning.	[Clark, Clark, 1977, p. 444]
Idioms	"Idioms are phrases with special meanings. Note that 'kick the dog' gets its sense from kick ('strike with ones foot') and dog ('canine animal'), for it means 'strike the canine animal with one's foot.' Not so with the idiom 'kick the bucket in the sense of 'die'. It does not mean 'strike the pail with ones foot.' to be represented, the phrase 'kick the bucket' must have a lexical entry all its own, one unrelated to those for kick and bucket. It is also true of other idioms, such as 'hit the sack', 'put one's foot in one's mouth', or be in hot water."	[Clark, Clark, 1977, p. 446]
Intentionality	"What a sentence means (a proposition) is often quite different from what we mean by it, which is sometimes a statement, sometimes a warning, sometimes a request, and sometimes something else."	[Odell, 1981] in [Sowa, 1984,p. 341]
Interpretation, Meaning	"The totality of meaning is never fully rendered: there is an immense amount of implications, even in the most explicit of languages; or rather nothing is ever completely expressed, nothing exempts the subject who is listening from taking the initiative in giving an interpretation."	[Merleau-Ponty, 1973, p. 29]
Lexical Creativity	"People are not content to leave language alone. When it leaves them too little room to maneuver efficiently, they invent new uses for old words and sometimes invent new words altogether. Some innovations are so natural that they go unnoticed: This mount is jeepable, The player had to be stretchered off the filed; and The rocket faltered at lift-off. ... The mental lexicon does not contain them ready-made in this form. They must be created on the spot as a normal part of speaking and listening."	[Clark, Clark, 1977, pp. 446-447]
Mandatory Inheritance	"Mandatory inheritance of properties is too inflexible for representing real-world knowledge[Fox, 9179]. The real world contains exceptions to almost every generalization."	[Touretzky, 1986, p. 3]
Manifolds	"A manifold is a set of at least n distinct elements for some chosen greater than 1." This is usually specified when the terms 'many' or 'nearly all' are used. "Could a logic of 'many' and 'nearly all' provide a semantics for inheritance systems, Unfortunately no, ... but Non-monotonic logics offer a solution to this problem."	[Touretzky, 1986, pp. 23-26]
Metaphors	Example of a metaphor: Esther Williams is a fish. (false) Esther Williams is a regular fish. (true)	[Lackoff, 1987, p. 138]
Metonymy	"Is the process of referring to something by the name of something else associated with it."	[Sowa, 1991, p. 10]
Multiple Category Membership	"... the role of multiple category membership and the degree of membership to a category is difficult to incorporate into taxonomic structures. Sometimes, whether or not there should be single class is obscure."	[Mazlack]

Nonfunctionality	"What a given string of words means is not a function of the formal characteristics those strings possess. 'Why not?' can be used to make a request, even though its form is that of a question."	[Odell, 1981] in [Sowa, 1984,p. 341]
Open texture	"Even if we legislate sets of necessary and sufficient conditions to govern what they mean, we can't be sure that our legislations will preclude the existence of contexts where we will be uncertain what our words mean, that is, we can still imagine cases where we wouldn't know whether or not a given word applied."	[Odell, 1981] in [Sowa, 1984,p. 342]
Overlapping and criss-crossing definitions	"Since most of the general empirical terms of a natural language are family resemblant in nature, it follows that in order to get at their meanings, i.e. the concepts they express, one must specify the set of overlapping and criss-crossing characteristics which determine the similarities and differences relevant to the question of whether or not some imagined or existing case falls under the concept in question."	[Odell, 1981] in [Sowa, 1984,p. 342]
Polysemy	"They have more than one sense."	[Clark, Clark, 1977, p. 444]
Primitive based theories	"Theories based on primitives have the following weaknesses: 1) No linguistic or psychological evidence has uncovered a truly universal set of primitives." Even Aristotle used different categories in different writings, never giving one final set of primitives.	[Sowa, 1984, p. 14]
Redundant Links	Representation: Inheritance Systems "Redundant links can confuse inheritance reasoners that use a path length ordering rather than inferential distance."	[Touretzky, 1986, p. 212]
Sincerity	"A very large number of speech acts which can be implemented in a natural language involve expressing one's emotions. A natural language incorporates the distinction between a genuine and a non genuine expression of an emotion. Expressing concern and expressing genuine concern are recognizably quite different."	[Odell, 1981] in [Sowa, 1984,p. 342]
Size	"We believe that a conservative estimate of the 'rules' and 'facts' required to encode all relevant aspects of the domain of common sense will easily run into the tens of millions - perhaps even more."	[Shastri, 1991, p. 113]

Appendix N: Conceptualization Comments

Topic	Fact	Reference
Ambiguity	"Many investigators have proposed the **Garden Path Theory**, or One meaning, Theory of ambiguity. As listeners proceed through a sentence they compute only one reading for each ambiguous construction. ... Only if this reading later becomes implausible or contradictory do they go back and compute a second, third, or fourth interpretation."	[Clark, Clark, 1977, p. 80]
Ambiguity	"The **Many Meanings Theory** claims that listeners compute two or more readings for each ambiguous construction and the immediately pick one on the basis of context."	[Clark, Clark, 1977, p. 81]
Ambiguity	"By the **No Meaning Theory**, on the other hand, listeners compute no meaning for an ambiguous construction at first, but let the context 'give weight' to one reading until it pops out."	[Clark, Clark, 1977, p. 81]
Ambiguity	Mixed Theory "(1) When listeners encounter an ambiguous construction, they compute multiple readings. (2) Using the context, listeners then attempt to select the most plausible reading. (3) if the ambiguity has not been resolved by the end of the clause, they select one reading and stick to it. (4) If later context contradicts the selected reading, they try to retrieve the surface structure of the prior clause and compute a new compatible reading."	[Clark, Clark, 1977, p. 82]
Basic-Level	"Basic-Level categories have an integrity of their own. They are our earliest and most natural form of categorization."	[Lackoff, 1987, p. 49]
Basic-Level, Prototypes	One of the main claims of the book "Women, Fire, and Dangerous Things" is that "language makes use of our general cognitive apparatus. If this claim is correct, tow things follow: - Linguistic categories should be of the same type as other categories in our conceptual system. In particular, they should show prototype and basic-level effect."	[Lackoff, 1987, p. 58]
Basic-Level	"As [Zubin and Kopcke, 1986] observe, the German gender system is governed by a number of semantic principles. One important principle i s based on the distinction between superordinate and basic-level categorization. In general superordinates are neuter, while basic-level concepts are typically either masculine or feminine." Lackoff goes on to say that basic-level concepts are cognitively simple, easy to process by humans (easy to remember, use and learn).	[Lackoff, 1987, p. 199]
Basic-Level, Kinesth image-schematic structure	1) There are at least two kinds of structure in our preconceptual experiences. A) Basic-Level Structure: Basic-Level categories are defined by the convergence of our gestalt perception, our capacity for bodily movement, and our ability to form rich mental images. B) Kinesth image-schematic structures: Image schemas are relatively simple structures that constantly recur in our everyday bodily experience: Containers, Paths, Links, Forces, Balance, and in various orientations and relations: Up-Down, Front-Back, Part-Whole, Center-Periphery, etc. These structure are directly meaningful, first because they are directly and repeatedly experienced because of the nature of the body and its mode of functioning in our environment."	[Lackoff, 1987, pp. 267-268]

Basic-Level	"Basic-Level category structure reflects the bodily nature of the people doing the categorizing, since it depends in gestalt perception and motor movements."	[Lackoff, 1987, p. 371]
Basic-Level	"Basic-Level structure - is partly characterized by human imaginative processes: the capacity to form mental images to store knowledge at a particular level of categorization and to communicate."	[Lackoff, 1987, p. 371]
Basic-Level	"Basic-level categories are especially easy for children to learn; superordinates are especially difficult (Markman & Callanan, 1984). Accordingly, the basic level may have special status for promoting children's inferences.	[Gelman, 1988, p. 68]
Basic-Level	"Second graders also relied on category level and property generalizability to guide induction. They, too, drew fewer inferences to superordinate or unrelated than basic level categories."	[Gelman, 1988, p. 85]
Basic-Level	"Basic categories are the categories for which the cue validity of attributes within categories is maximized since superordinate categories have fewer common attributes within the category than do basic categories and subordinate categories share more attributes with contrasting categories than do basic categories. Basic categories are, thus, the categories which mirror the correlation structure of the environment."	[Rosch, Mervis, 1975, p. 602]
Basic-Level	The author argues the existence of inherently neutral specificity (Basic-Level items). Sample Question for Basic Level determination: *What have you got in your pocket?* *a) An apple* *b) A Granny Smith* *c) A fruit*	[Cruse, 1975, p. 153]
Basic-Level	"Categorizations which humans make of the concrete world are not arbitrary but highly determined. In taxonomies of concrete objects, there is one level of abstraction at which the most basic category cuts are made. Basic categories are those which carry the most information, possess the highest category cue validity, and are, thus, the most differentiated from one another. ... basic objects are the most inclusive categories whose members: (a) possess significant numbers of attributes in common, (b) have motor programs which are similar to one another, (c) have similar shapes, and (d) can be identified from averaged shapes of members of the class."	[Rosch, Mervis, 1976, p. 382]
Basic-Level	"Children's initial categories are basic-level categories. This seems reasonable, because ... , categories at the basic level are more fundamental psychologically than categories at other taxonomic levels. For example, chair (basic-level category) is more fundamental than either kitchen chair (a subordinate category) or furniture (a superordinate category). Categories at the basic level 'stand out' as categories. Theses categories are based on large clusters of (subjectively) correlated attributes that overlap very little from category to category. In our world, these basic-level categories are the most general categories whose members share similar overall shapes (or similar parts in particular configurations; and similar functions or characteristic actions. Recent research has indicated that two-year-olds can form basic-level categories easily, while formation of superordinate and subordinate categories is considerably more difficult.(Daehlr, Lonardo, & Bukatoko 1979; Mervis & Crisafi, 1982)."	[Mervis, 1987, p. 202]
Basic-Level	The authors discuss a folk-lore category level equivalent to our basic-level categories among the Tzeltal peoples based upon motor actions, physical appearance, imagination, and cultural significance; it is call the folk generic taxa.	[Berlin, Breedlove, Raven, 1974, pp. 25-45]

Basic-Level Categorization	"Denny's observation fit nicely with observations by Berlin, Rosch... on basic-level categorization. What they found was that basic-level categorization depended on the nature of everyday human interaction both in a physical environment and in a culture. [Berlin, Breedlove, and Raven 1974] [Rosch 1977]. Factors involved in basic-level categorization include gestalt perception, motor interaction, mental images, and cultural importance. Taken together, these observations support the view that our conceptual system is dependent on, and intimately linked to, our physical and cultural experience. It disconfirms the classical view that concepts are abstract and separate from human experience."	[Lackoff, 1987, pp. 112-113]
Basic-Level Examples	Nouns - Chairs, tables, houses, books, lamps, coats, cars, etc. Actions - running, walking, eating, drinking, etc. Properties - tall, short, hard, soft, heavy, hot, cold, etc. Colors - black, white, red, green, blue, and yellow	[Lackoff, 1987, pp. 270-271]
Categories	George Lackoff says "This book attempts to bring together some of the evidence for the view that reason is embodied and imaginative - in particular, the evidence that comes from the study of the way people categorize." He also explains that conceptual systems are organized in terms of categories, and most if not all of our thought involves those categories.	[Lackoff, 1987]
Categories	Rosch and others have observed that categories in general have best examples (called "prototypes") and that all of the specifically human capacities like body movement, perception, mental image formation, learning, memory, and other body functions play a role in categorization.	[Lackoff, 1987, p. 7]
Categories	"The main thesis of this book is that we organize our knowledge by means of structures called idealized cognitive modes, or ICMs, and that category structure and prototype effects are by-products of that organization."	[Lackoff, 1987, p. 68]
Categories	"Our concept of reality is a matter of our linguistic categories."	[Sowa, 1984, p. 346]
Categories	"I am saying that what counts as reality - what counts as a glass of water or a book or a table, what counts as the same glass or a different book or two tables - is a matter of the categories that we impose on the world; and those categories are for the most part linguistic. And furthermore; when we experience the world we experience it through linguistic categories that help to shape the experiences themselves. the world doesn't come to us already sliced up into objects and experiences: what counts as an object is already a function of our system of representation, an how we perceive the world in our experiences is influenced by that system of representation. .... Our concept of reality is a matter of our linguistic categories."	[Searle, 1978, p. 184]
Categorization, Prototypes	"Category structure plays a role in reasoning. In many cases, prototypes act as a cognitive reference points of various sorts and form the basis for inferences [Rosch 1975a, 1981]. The study of human inference is part of the study of human reasoning and conceptual structure; hence, those prototypes used in making inferences must be part of a conceptual structure."	[Lackoff, 1987, p. 45]
Categorization	"Categorization is the simplest form of restructuring."	[Sowa, 1984, p. 332]

Categorization	"Most would agree that people use concepts both to provide taxonomy of things in the world and to express relations between classes in that taxonomy. The taxonomic function can itself be split into the following two generally agreed upon component functions: 1. Categorization - This function involves determining that a specific instance is a member of a concept(i.e. this particular creature is a guppy) or that one particular concept is a subset of another (i.e. guppies are fish). 2. Conceptual combination ...."	[Smith, Medin, 1981, p. 7]
Categorization	"Young children use category membership to predict underlying similarities among objects - even when perceptual similarity would lead to a different prediction."	[Gelman, 1988, p. 66]
Categorization	"Since the time of Aristotle, philosophers have argued that our conceptual system is articulated by a core of ontologically basic categories, such as physical object and event. Within the category of physical objects, living thing is such a category, as are animal and plant within the category of living things."	[Carey, 1985, p. 162]
Categorization	"Man is, by nature, a classifying animal. His continued existence depends, in fact, on his ability to recognize similarities and differences among objects and event in his physical universe and to mark these similarities and differences linguistically. "	[Berlin, Breedlove, Raven, 1974, p. 25]
Categorize	"To categorize is to render discriminatably different things equivalent, to group objects and events and people around us into classes, and to respond to them in terms of their class membership rather than their uniqueness."	[Bruner, Goodnow, Austin, 1956, p. 1]
Classical Categories	Rosch says "... many cognitive models use classical categories... since they are a product of the human mind."	[Lackoff, 1987, p. 160]
Classical Categorization Criticisms	"1) The classical view only deals with structural features - fixed properties that describe an entity in isolation, like the handle or concavity of a cup - and prohibits functional features, like the fact that a cup is used to hold something. 2) The classical view excludes disjunctive concepts (i.e. two definition for the same concept). 3) The classical view assumes that if concept X is a subset of concept Y, the defining features of Y are nested in those of X. 4) The heart of classical view is its assumption that every concept has a set of necessary and sufficient features."	[Smith, Medin, 1981, pp. 26-32]
Clustering	Clustering techniques are best suited for numerical data.	[Fisher & Pazzani, 1991, p. 15]
Communicating	[Grice, 1975] "Cooperative Principle 4 Basic Maxims - Quantity: Say neither too much nor to little. - Quality: Try to make your contribution one that is true. - Relation: Be relevant. - Manner: Be perspicuous. ... sometimes the 4 maxims are not followed when people are joking, being evasive, or lying."	[Sowa, 1984, p. 266-267]
Concept Formation	"A concept formation system accepts a stream of observations (i.e. events, objects, instances), and discovers a classification scheme over the data stream."	[Fisher & Pazzani, 1991, p. 3]
Concepts	"Concepts are inventions of the human mind to construct a model of the world."	[Sowa, 1984, p. 344]

Conceptualizing Capacity	"- The ability form symbolic structures that correlate with preconceptual structures in our everyday experience. Such symbolic structures are basic-level and image schematic concepts. - The ability to project metaphorically from structures in the physical domain to structures in abstract domains, constrained by other structural correlation between the physical and abstract domains. this accounts for our capacity to reason about abstract domains such as quantity and purpose. - The ability to form complex concepts and general categories using image schemas as structures and taxonomies with superordinate and subordinate categories."	[Lackoff, 1987, pp. 280-281]
Cue Validity	"Mathematically, cue validity has been defined as a conditional probability - specifically, the frequency of a cue being associated with the category in question divided by the total frequency of that cue over all relevant categories."	[Rosch, Mervis, 1975, p. 575]
Domain Specific	Norm Sounheimer & Rich Thomason both advocate there is merit in working with micro systems (domain specific).	[Sowa, 1991, pp. 34-36]
Embodiment	"Suppose a child is born devoid of all senses; he has no sight, no hearing, no touch, no smell, no taste - nothing. There's no way whatsoever for him to receive any sensations from the outside world. As suppose the child is fed intravenously and otherwise attended to and kept alive for eighteen years in this state of existence. The question is then asked: Does this eighteen-year-old person have thought in his head? If so, where does it come from? How does he get it? The philosopher Hume would have answered that the eighteen-year-old had no thoughts whatsoever..."	[Pirsig, 1974, pp. 123-124]
Entity-Relationship Models	"... In another approach, which has been adopted, e.g., by the expert system PLAKON, variant relationships are introduced by differentiating between AND branches for the usual part_of decomposition and OR branches representing choices among the items listed. However, such an approach allows only one kind of branching per nesting level, thus requiring auxiliary nodes for every variant. To the best of our knowledge, there is no model allowing to mix AND and OR branches."	[Zhou & Baumann, 1992, p. 25]
Family Resemblance	"A family resemblance relationship consists of a set of items of the form AB, BC, CD, DE. That is, each item has at least one, and probably several, elements in common with one or more other items, but no, or few elements are common to all items."	[Rosch, Mervis, 1975, p. 575]
Family Resemblance and Prototype Effects	"In the present research, we viewed natural semantic categories as networks of overlapping attributes; the basic hypothesis was that members of a category come to be viewed as prototypical of the category as a whole in proportion to the extent to which they bear a family resemblance to (have attributes which overlap those of) other members of the category. Conversely, items viewed as most prototypical of one category will be those with least family resemblance to or membership in other categories."	[Rosch, Mervis, 1975, p. 575]
First Order Logic	"First order logic is by far the most widely used formal representation language, yet much of the knowledge that interest AI. researchers cannot at present be conveniently represented in first order logic."	[Touretzky, 1986, p. 215]
Hierarchies	"A series of psychological studies [Kintsch 1974] [van Dijk 1975] [Meyer 1975] [Thorndike 1977] and [Crothers 1972]) offers strong evidence for the hypothesis that a person, after reading a text, has developed a macro structure that organizes the propositions of the text in a hierarchical form. The propositions highest in the hierarchy are more general than those that occur at the lower levels."	[Simmons, Correira, 1979, p. 363]

Hierarchy	"... And you see that every time I made a further division, up came more boxes based on these divisions until I had a huge pyramid of boxes. Finally you that while I was splitting the cycle up into finer and finer pieces, I was also building a structure. This structure of concepts is formally called a hierarchy and since ancient times has been a basic structure for all Western knowledge. Kingdoms, empires, churches, armies have all been structured into hierarchies. Modern business are so structured. Tables of contents of reference material are so structured, mechanical assemblies, computer software, all scientific and technical knowledge is so structured - so much so that in some fields such as biology, the hierarchy of phylum-order-class-genus-species is almost an icon."	[Pirsig, 1974, p.93]
Incremental Learning, Categorization	In talking about incremental learning the following was written. "However in many cases human learners appear to assimilate instances as they become available. We refer to this process - the incremental unsupervised acquisition of categories and their intentional descriptions - as concept formation."	[Fisher & Pazzani, 1991, pp. 22 -23]
Incremental Learning	Incremental Process - "This means that learning occurs with each instance, and that the system does not need to reprocess all previously seen examples in order to learn. This is a fundamental constraint imposed by psychological results...."	[Iba & Gennari, 1991, p. 358]
Inheritance Hierarchies	"We all know the world is not made from type hierarchies (ISA Hierarchies) only, and we need many different types of relationships to model the complex world."	[Chen, 1992, p. 1]
Inheritance Systems	"The prime reason for omitting explicit quantification from inheritance systems is that it would require a much more complicated inference algorithm. Simple graph-traversal inference algorithms are one of the major attraction of inheritance systems."	[Touretzky, 1986, pp. 199-200]
Inheritance Systems	"Oh, woe to the AI hacker who insists that a system do the right thing all of the time."	[Touretzky, 1986, p. 208]
Knowledge Base	"In order to build a knowledge base of any complexity, it is necessary to lay a foundation of basic concepts and define other in terms of them. These defined concepts are then used to define still other concepts until a substantial knowledge system is built up from many layers of defined abstractions."	[Woods, 1991, p. 63]
Language	"As yet, no program can accept the full range of human language and interpret all the nuances and shades of meaning that people normally use."	[Sowa, 1984, p. 286]
Learning	"- First children associate words with the concrete concepts used in perceiving the world and acting upon it. - Next they learn syntactic rules for mapping concepts and conceptual relations to well-formed sentences. - Finally, they master the formal structures of the type hierarchy and the fine points of syntax."	[Sowa, 1984, p. 215]
Learning	"Conceptual analysis enables a systems analyst to define new concepts and schemata for a knowledge based system. People, however, learn such things from experience. The ideal knowledge acquisition tool would be a computer system that could learn concepts and schemata by itself, either from experience or from reading the same kinds of dictionaries and encyclopedias that people read. The system could be primed with a small, but general vocabulary of about 2,000 words, the concepts and conceptual relations needed for defining those words, and enough syntactic and semantic rules to enable it to read a book to learn more."	[Sowa, 1984, p. 329]

Machine Translation	"dismissed the possibility that a computer could ever do high quality machine translation: what such a suggestion amounts to, if taken seriously, is the requirement that translation machine should not only be supplied with a dictionary but also a universal encyclopedia." [Bar-Hillel, 1960]	[Sowa, 1984, p. 127]
Meaning, Communicating	"The meaning of a sentence is only partly determined by the meaning of words. A large, if not the major part of meaning comes from the context the speaker's intentions, and listener's expectations."	[Sowa, 1984, p. 264]
Meaning, Communicating	"In normal conversations, the participants have common air: to make themselves understood and to understand the other speaker... But at the same time people don't want to waste their own efforts or let the listeners get bored. Therefore, they omit everything that they believe is common knowledge."	[Sowa, 1984, p. 266]
Metaphors	"A standard catalog of metaphors is probably adequate for interpreting everyday speech."	[Sowa, 1984, p. 271]
Metaphors	"Children's speech has an extreme literalness and absence of metaphor."	[Chukovsky, 1963, p. 271]
Parsing	"Many parsers are god enough for practical purposes, but none of them have attained the ideal."	[Sowa, 1984, p. 248]
Proper Names and Common Nouns	"Let us take it that children as young as 17 months are able to distinguish between common and proper nouns."	[Macnamara, 1982, p. 29]
Prototypes	"However, the present study has shown that empirically defined prototypes of natural categories are just those items with highest cue validity. Such a structure of categories would, in fact, appear to provide the means for maximally efficient processing of categories."	[Rosch, Mervis, 1975, p. 601]
Prototypes	"The present study has shown that formation of prototypes of categories appears to be likewise non arbitrary. The more prototypical a category member, the more attributes it has in common with other members of the category and the less attributes in common with contrasting categories. Thus, prototypes appear to be just those members of the category which most reflect the redundancy structure of the category as a whole."	[Rosch, Mervis, 1975, p. 602]
Reference Point (Best Example)	"To be a 'reference point' within a category , a stimulus must be shown to be one which other stimuli are seen 'in relation to.'	[Rosch, 1975, p. 532]
Reference Point (Best Example)	"'**A _____ is (almost, virtually, essentially, etc.) a _____.**' was presented to subject who were provided with two stimuli: the prototype and a stimulus slightly deviant from the prototype (e.g., a 'pure' focal red and an 'off' red). The subjects' task was to place one stimulus in each of the blanks. control pairs were stimuli neither of which were prototypes. if the relation between the presumed reference stimulus and the deviant stimulus were symmetrical the stimuli should be placed in either blank equally often. The hypothesis, however, was that deviations would be more often placed in the first blank and reference stimuli in the second."	[Rosch, 1975, p. 533-534]
Semantic Network	Shastri, in this paper says semantic nets are used because they are computational effective in knowledge representations and reasoning where inference algorithms are used.	[Shastri, 1991, p. 113]
Semantic Network	"Reasoning that may be characterized as inheritance and recognition (classification) within a semantic network plays a central role in language understanding, visual recognition, and commonsense reasoning. Inheritance and recognition are also significant because humans can perform these inference effortlessly and extremely fast - to wit language understanding in real time."	[Shastri, 1991, p. 120]

Spanning	*Spanning* - is when a predicate can be applied sensibly to a term. i.e. "The fight was bloody." *Category Mistake* - is that some predicates cannot be sensibly applied to some terms. i.e. "The fight was skinny."	[Carey, 1985, pp. 162-165]
Syntax	"Syntax is the stratum that combines words into sentences."	[Sowa, 1984, p. 218]
Taxonomies	"... in the third century A.D., the Greek philosopher Porphyry clarified the relationships among Aristotle's categories by drawing them in a tree. ... The tree of Porphyry was drawn by logicians in the middle ages. ... Porphyry's trees formed a taxonomic system or T-box for organizing concepts into types and subtypes."	[Sowa, 1991, pp. 13-14]
Taxonomy	"There is a common idea, at least in the west, and I would suspect in other cultures, that there is a single correct taxonomy of natural things - plants, animals, minerals, etc. A taxonomy is a cognitive model of a particular kind. Taxonomic models are common in cognition and they are built into languages through out the world. They are the most common means that human beings have used to make sense of there experience."	[Lackoff, 1987, pp. 118-119]
Taxonomy	"... In an ideal world, there would be no conflict among the three schools - cladistics, phenetics, and tradition and all would produce the same classification for a given set of organisms." Pheneticists look at the overall similarity in form, function, and biological role. Cladists are primarily concerned with branching order in the course of evolution and shared derived characteristics.	[Gould, 1983, pp. 363-364]

Appendix O: Complete Machine Categorization Results

Bear Article Categorization

```
entity
   America#0
   Europe#0
   Asia#0
      East#0 Asia#0
         South#0 East#0 Asia#0
   Alaska#0
   North America#0
   bear#0
      brown bear#0
         biggest#0 brown bear#0
            kodiak bear#0
         fiercest#0 brown bear#0
            grizzly bear#0
      blue bear#0
      black bear#0
         moon bear#0
      mother#1 bear#0
      spectacled bear#0
      sloth bear#0
         mother#1 sloth bear#0
      smallest#0 bear#0
         sun bear#0
      polar bear#0
   winter#0
   country#0
   hole#0
      den#0
   cave#0
      den#0
   baby#0
      cub#0
   midwinter#0
   mother#0
   springtime#0
   wood#0
   landmark#1
   food#0
   camper#0
   picnicker#0
   thing#0
      sweet#0 thing#0
   mountain#0
   mark#0
      white#0 mark#0
   fruit#0
   vegetable#0
   animal#0
   honey#0
   nest#0
      bee's#0 nest#0
         wild#0 bee's#0 nest#0
   sting#0
   mind#0
   plant#0
```

```
South America#0
insect#0
   tiny#0 insect#0
      termite#0
India#0
back#1
climber#0
arctic#0
land#0
ice#0
   floating#0 ice#0
year#0
snow#0
sea#0
berry#0
summer#0
   arctic#1 summer#0
      short#0 arctic#1 summer#0
eater#0
   flesh#0 eater#0
      polar bear#0
seal#0
   young#0 seal#0
fur#0
   white#0 fur#0
swimmer#0
   polar bear#0
family#0
```

Whale & Dolphin Article Categorization

```
entity
   fin#0
   sea#0
   fish#0
   mammal#0
      whale#0
         toothed#0 whale#0
            largest#0 toothed#0 whale#0
               sperm whale#0
            orca whale#0
               killer whale#0
         toothless#0 whale#0
         blue whale#0
         humpback whale#0
         sperm whale#0
         small#0 whale#0
      dolphin#0
         toothed#0 dolphin#0
         toothless#0 dolphin#0
   surface#0
   water#0
   air#0
   member#0
   animal#0
   family#0
   porpoise#0
      toothed#0 porpoise#0
      toothless#0 porpoise#0
   tooth#0
      peg-leg#0 tooth#0
      spade-like#0 tooth#0
   food#0
   hour#0
   elephant#0
   mile#0
   sound#0
   back#1
   human#0
      friendly#1 human#0
   plate#0
      horny#0 plate#0
   baleen#0
   bristle#0
   shrimp#0
   song#0
      special#0 song#0
   leap#2
      breach#0
   skin#0
      spongy#0 skin#0
   fat#0
      blubber#0
   calf#0
      whale#1 calf#0
   tail#0
```

```
mother#0
route#0
year#0
stay#2
kind#0
herd#0
people#0
diver#0
beach#0
foot#0
ton#0
world#0
shark#0
sea lion#0
victim#0
whaler#0
   Spanish#0 whaler#0
      early#0 Spanish#0 whaler#0
head#0
   beaked#0 head#0
   rounder#0 head#0
country#0
```

Spider Article Categorization

```
entity
   leg#0
   eye#0
      sharp#0 eye#0
   arachnid#0
      spider#0
         house spider#0
         garden spider#0
         hammock spider#0
         male#0 spider#0
         malmignette#0
            female#1 malmignette#0
         black widow#0
            American#0 black widow#0
         mother#1 spider#0
         wolf spider#0
         trapdoor spider#0
            funnel-web#0
         water spider#0
         jumping spider#0
            zebra spider#0
               little#0 zebra spider#0
         crab spider#0
         bird-eating spider#0
   insect#0
   creature#0
   food#0
   fly#0
   cobweb#0
   web#0
      spiders'#0 web#0
      funnel-shaped#0 web#0
      spider's#0 web#0
         water#2 spider's#0 web#0
   silk#0
   spinneret#0
   wire#0
      steel#0 wire#0
         stronger#0 steel#0 wire#0
   thread#0
      corner#0 thread#0
   gossamer#0
   breeze#0
   twig#0
   leave#2
      corner#0 leave#2
   spoke#0
   wheel#0
   center#0
   frame#0
   spiral#1
      main#0 spiral#1
         stick#0 main#0 spiral#1
   outside#0
   middle#0
```

```
vibration#0
poison#0
fang#0
juice#0
victim#0
female#0
message#0
lunch#0
   garden spider#0
mate#0
human#0
bite#0
egg#0
mother#0
cocoon#0
egg-sac#0
spiderlings#0
back#1
baby#0
America#0
burrow#0
   home#0
wait#0
entrance#0
Australia#0
child#0
   Australian#0 child#0
water#0
bubble#0
air#0
minnow#0
   tiny#0 minnow#0
bell#0
   diving#0 bell#0
weed#0
   water#1 weed#0
winter#0
line#0
falling#0
flower#0
butterfly#0
bee#0
enemy#0
animal#0
wasp#0
maggot#0
one#2
two#1
```