Towards Informal Computer Human Communication:
Detecting Humor in Restricted Domain

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

in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

In the Department of
Electrical and Computer Engineering
of College of Engineering

October 2008

By

Julia Michelle Taylor

M.S. Computer Science, University of Cincinnati, August 2004
B.S. Computer Science, University of Cincinnati, June 1999
B.A. Mathematics, University of Cincinnati, June 1999

Advisor and Committee Chair: Dr. Lawrence J. Mazlack
ABSTRACT

The dissertation presents a computational humor detector designed to look at any one of 200 short texts (100 jokes and 100 non-jokes) and to determine whether the text is a joke or not. This is accomplished on the basis of meaning recognition by the computer with the help of an ontology crafted from a children's dictionary, and any additional background knowledge necessary for text understanding. The research is underlaid by an advanced formal semantic theory of humor, and it constitutes the first known attempt to validate a theory of humor computationally. The results of the computational experiments are quite encouraging.

With the advancement of computational technologies, increasingly more emphasis continues to be placed on systems that can handle natural language, whether it involves human-computer communication, or comprehension of written narratives, information on the Web, or human conversations. Humor occurs frequently in verbal communication. Thus, without humor detection no natural language computer system can be considered successful. For full computational understanding of natural language documents and for enabling intelligent conversational agents to handle humor, humor recognition is necessary or at the very least highly desirable.

This exploratory research had to be constrained and its goals narrowed down for the purpose of implementability. The joke detector is therefore restricted to the recognition of short jokes. The domain is further restricted to jokes that are based on ambiguous words, where the detection of several meanings results in humor; and to jokes that are based on similar- or identical-sounding words, where the detection of correct pairs also leads to humor. Because of the meaning-based nature of the research, the system can be extended to other types of humor in text, without changes to the algorithm.
The central hypothesis is that humor recognition of natural language texts is possible when the knowledge needed to comprehend the texts is available in a machine-understandable form. To test the hypothesis, a description logic ontology was built to represent knowledge manifested in natural language texts. The results show that when the information, necessary for humans to understand humor, is available to a machine, it successfully detects humor in text.
Copyright or blank page
ACKNOWLEDGEMENTS

I am grateful to the many people who provided useful comments at many conferences and workshops as well as in classes, taken and taught, and whose questions led to many ideas, some implemented and some still in progress. I would like to thank Drs. Peter Chiu and Joseph Foster, whose classes in psychology and linguistics, respectively, were very helpful for understanding the complexity of the human brain and language. Special thanks go to Dr. Christian Hempelmann for his very helpful comments on joke analysis and to Dr. Victor Raskin for making his research, experience, and commentary available to me.

This work has benefited greatly from discussions with my Dissertation Advisory Committee members, Drs. Carla Purdy, John Schlipf, Peter Chiu and Michèle Vialet. My sincere thanks go to them for making themselves available and for being interested and patient, all when I needed their assistance the most.

I would like to express my special gratitude to Dr. Lawrence Mazlack, my advisor, who made this unusual project not only possible but also very enjoyable. His intuition, support, patience, suggestions, and many late evening sessions are only a few from a very long list of his generous contributions to this work.

Last but not least, I am grateful to my husband, Matthew Brian Taylor, and my entire family, without whose love, help, understanding, support, and endless patience it would be much harder for me to finish. Matthew’s invaluable insight into humor from a practitioner’s point of view supplemented my work in a unique way.

This work has been partially supported by an OBR Distinguished Doctoral Research Fellowship in Computer Science and Engineering and a Rindsberg Fellowship, for which I am ever thankful.

iv
# Table of Contents

1 Introduction.......................................................................................................................... 1

1.1 Goals of This Research ................................................................................................. 1

1.2 The Difficulties of Computational Humor Detection ...................................................... 3

1.3 Motivation for Research in Computational Humor .......................................................... 4

1.4 Roadmap .......................................................................................................................... 7

2 Background .......................................................................................................................... 10

2.1 Natural Language ............................................................................................................ 11

2.1.1 Sentence Meaning and Semantic Roles .................................................................... 14

2.1.2 Ambiguity in Natural Language ................................................................................ 19

2.1.3 Detecting Humor and Statistical Language Processing ............................................ 22

2.2 Lexical Access and Semantic and Phonological Associations and Priming ................... 25

2.2.1 Word Frequency ......................................................................................................... 28

2.2.2 Lexical Ambiguity ........................................................................................................ 29

2.2.3 Orthographic Effects .................................................................................................. 30

2.2.4 Phonological Effects .................................................................................................. 31

2.2.5 Semantic Effects ......................................................................................................... 32

2.3 Knowledge Representation .............................................................................................. 33

2.3.1 Semantic Networks ..................................................................................................... 34

2.3.2 Conceptual Graphs ...................................................................................................... 35

2.3.3 Frames ........................................................................................................................ 36

2.3.4 Conceptual Dependencies and Scripts ...................................................................... 39

2.3.5 Description Logics ........................................................................................................ 43

2.3.6 Ontologies .................................................................................................................... 45
LIST OF FIGURES

Figure 2-1: Different parse trees for Text 11.................................................................21
Figure 2-2: An example of semantic network (Baader, 1999)........................................35
Figure 2-3: An example of conceptual graph (Sowa, 2000)...........................................36
Figure 2-4: Example of a frame (Luger & Stubblefield, 1998).......................................37
Figure 2-5: Frames in FrameNet (Baker, Fillmore, & Lowe, 1998)...............................39
Figure 2-6: Restaurant script, scenes Enter and Order (Luger & Stubblefield, 1998).......40
Figure 2-7: Restaurant script, scenes Eating and Exiting (Luger & Stubblefield, 1998)....41
Figure 2-8: Components of a Restaurant script (Luger & Stubblefield, 1998)...............41
Figure 2-9: Syntax and semantics of some constructors (Horrocks, 2005)......................43
Figure 2-10: TBox example (Baader & Nutt, 2003).....................................................43
Figure 2-11: ABox example (Baader & Nutt, 2003)......................................................44
Figure 2-12: Description Logics expressivity ....................................................................44
Figure 2-13: Top level ontology (Sowa, 2000).................................................................46
Figure 2-14: Types of humor theories (Attardo, 1994)....................................................47
Figure 2-15: Script arrangement (Raskin, 1985)...............................................................50
Figure 2-16: Script overlap and opposition (Attardo, Hempelmann, & Di Maio, 2002)....52
Figure 3-1: Humor detector architecture............................................................................67
Figure 3-2: Semantic role labeling output for Text 6 and Text 8 from a demo system (Semantic Role Labeling Demo)..............................................................................74
Figure 3-3: A parse tree used to select semantic roles for Text 6..................................75
Figure 3-4: A parse tree used to select semantic roles for Text 8..................................75
Figure 3-5: Semantic role labeling output for Text 24 from the demo system.................76
Figure 3-6: A parse tree for Text 24 from the demo system ..............................................76
Figure 3-7: Disjointness of concepts through their properties ..........................................80
Figure 3-8: Number of word replacements ........................................................................86
Figure 4-1: Abstraction of humor detector........................................................................88
LIST OF TABLES

Table 2-1: Thematic roles (Finegan, 2004)..............................................................15
Table 2-2: Case roles (Nirenburg & Raskin, 2004)......................................................16
Table 2-3: Thematic roles (Jurafsky & Martin, 2000)......................................................16
Table 2-4: Pun vs. wordplay in terms of SO and LM (Hempelmann, 2003).....................55
Table 3-1: Dictionary definitions and their DL representation .....................................70
Table 3-2: Specific knowledge and its DL representation ..............................................71
Table 4-1: Categorization of jokes to be tested ..........................................................93
Table 4-2: Non-jokes, created from jokes by removing word ambiguity or phonological similarity.........................................................94
Table 5-1: Known words from The American Heritage First Dictionary (2006).............102
Table 5-2: Known meanings from The American Heritage First Dictionary (2006) .........102
Table 5-3: Salient meanings needed for the joke detection from The American Heritage First Dictionary (2006) .................................................................103
Table 5-4: Relationships between concepts needed for joke detection from The American Heritage First Dictionary (2006) .................................................................103
Table 5-5: Known words from The Dorling Kindersley Children’s Illustrated Dictionary (McIlwain, 1994) .................................................................104
Table 5-6: Known meanings that are needed for joke detection from The Dorling Kindersley Children’s Illustrated Dictionary (McIlwain, 1994) .................................................................104
Table 5-7: Salient meanings needed for the joke detection from The Dorling Kindersley Children’s Illustrated Dictionary (McIlwain, 1994) .................................................................105
Table 5-8: Relationships between concepts needed for joke detection from The Dorling Kindersley Children’s Illustrated Dictionary (McIlwain, 1994) .................................................. 105

Table 5-9: Text detection when all required information is known ........................................... 105

Table 5-10: Phonologically similar words ..................................................................................... 106

Table 5-11: Known words for phonological jokes from The American Heritage First Dictionary. .......................................................................................................................... 107

Table 5-12: Known meanings for phonological jokes from The American Heritage First Dictionary.......................................................................................................................... 107

Table 5-13: Salient meanings, needed for the phonological joke detection from The American Heritage First Dictionary .......................................................................................................................... 108

Table 5-14: Relationships between concepts, needed for phonological joke detection, from The American Heritage First Dictionary .......................................................................................................................... 108

Table 5-15: Text detection when all required information is known. ........................................... 109

Table 5-16: Joke detection with previously known jokes ............................................................. 111

Table 5-17: Subsumption vs. regular joke detector ...................................................................... 111

Table 5-18: Detection of texts according to their themes ............................................................. 111

Table 5-19: Detection of texts according to their themes ............................................................. 112
LIST OF REPEATING ACRONYMS

DL: Description Logics

GTVH: General Theory of Verbal Humor (Attardo & Raskin, 1991)

KR: Knowledge Representation

LA: Language (from the General Theory of Verbal Humor)

LM: Logical Mechanism (from the General Theory of Verbal Humor)

NP: Noun Phrase: a phrase whose head is a noun or a pronoun

NS: Narrative Strategy (from the General Theory of Verbal Humor)

N+V: Normality + Violation (Veatch, 1998)

PP: Prepositional Phrase

SI: Situation (from the General Theory of Verbal Humor)

SO: Script Overlap/Script Oppositeness (from the Script-based Semantic Theory of Humor and
the General Theory of Verbal Humor)

SSTH: Script-based Semantic Theory of Humor (Raskin, 1985)

TA: Target (from the General Theory of Verbal Humor)

VP: Verb Phrase: a phrase whose head is a verb
1 Introduction

Much of human knowledge is communicated through natural language, and much of this knowledge is stored in natural language texts. Computational understanding of natural language is critical to handling the large amounts of existing texts as well as the future’s geometrically growing volume of texts. Knowledge cannot be readily summarized or accessed unless inherent utterance ambiguity is recognized and resolved. While there are many existing successful disambiguation tools, intentional ambiguity recognition has not received much attention. Humor often depends on intentional ambiguity and offers a clear, distinct research focus. People use humor in their communication, and some natural language documents contain jokes and other forms of humor. Thus, humor must be detected to achieve an understanding of such documents.

1.1 Goals of This Research

This research is concerned with the computational detection of humor in text. Recognition of all humor in text is an overly broad task. The domain is narrowed for this dissertation. The goal of this work is to recognize wordplay-based humor in short texts written for young children. Specifically, the goal is to determine if humor detection is possible without following known precise templates. The focus is on the meaning of text, not its structure. The goal is a step toward overall computational humor recognition, a necessary step in computational understanding of natural language texts. The central hypothesis is that humor recognition of natural language texts is possible when the knowledge needed to comprehend those texts is available in a machine-understandable form. The hypothesis was tested by building an ontology, using description logic to represent knowledge manifested in natural language texts, and using
the ontology to detect humor in texts recognized as humorous by humans. The project aims at answering the following questions based on information in this ontology:

- Is it possible to recognize jokes that are based on lexical ambiguity?
- Is it possible to recognize jokes that are based on the phonological similarity of words?
- Can jokes be recognized by comparing them to already known jokes?
- Can jokes be recognized when an ontology does not have all of the required background knowledge to process the meaning of a text?
- Are some types of jokes easier to recognize than others?

Ontology and humor theories -- namely, the Script-based Semantic Theory of Humor (Raskin, 1985) and General Theory of Verbal Humor (Attardo and Raskin, 1991) -- help to answer these questions and to test the ideas and principles of computational detection and analysis of jokes and non-jokes.

There are several reasons for undertaking a computational humor-understanding project. First, full computational understanding of natural language documents is impossible without recognizing humor as some texts include jokes and other forms of humor. Second, humor detection and production are needed to achieve sociable computing, that is, to enable computers to socially interact normally with most people, not just with experts in computing.

Interest in sociable computing has been on the increase: from government and corporate users to technicians to general public, people want to make the computer a more human-like companion and collaborator. The best way for a sociable computer to interact with people is through informal, every day language. Humor is a special kind of informal communication that computers will need to be able to perform before non-technically trained people can interact with them, as naturally as they do with other people.
Additionally, according to Binsted et al. (2006), computational understanding of humor is a required step in achieving extensive human computer interaction because humor is such a fundamental part of human interaction that no computational intelligent conversational agent is complete without an ability to understand humor.

1.2 The Difficulties of Computational Humor Detection

Computational humor detection is not an easy task. To detect the same humor as that perceived by speakers of a given language, the computer must have:

- An understanding of the natural language used in a humorous text.
- A computational representation of knowledge that is accessed by the situations described in a text.
- An ability to reason about these situations.
- A humor theory suitable for algorithmic representation.

As an example, consider the following joke (Reilly, 2007):

Text 1: A man walks into a bar. Ouch.

For a person to recognize this as a joke, at the very least, he or she needs to know the meanings of each of the words in Text 1, including two meanings of the word *bar*. The person must understand what some of the word combinations mean, including two meanings of the phrase *walk into*. The person needs to access this information in such a way that both situations of “entering a bar” and “hitting a bar” are available. It is likely that the former situation will occur to the person first as the more common one. However, the word *ouch* will lead him or her to the latter.
For a computer to detect this joke, several components must be added to the criteria used in human detection. First, the knowledge that humans have must somehow be represented for a computer. Thus, a computer must have access to knowledge of the sort: people can move around; they can drink in bars; when people hit something, they feel pain; a man is a person. Second, once the computer has this “knowledge” it must be able to reason with it. For example, if person can move and man is person and walk is move, then man can walk. Third, a computer must have an algorithm for classification of humorous and non-humorous texts. Such classification could best be achieved based on humor theories.

Work on computational humor is multidisciplinary: it requires linguistics and psychology if one is interested in understanding and modeling human perception of humor, and computer science, if one needs to computationally model and detect humorous texts.

This dissertation concentrates on detection of jokes where the joke text is written but not spoken. Recognizing a written joke requires use of natural language and background knowledge, but is not complicated by the additional issues that are introduced by speech recognition.

Speech recognition has additional problems and challenges. These problems would have to be tackled before a spoken joke is converted into a text if spoken jokes were considered. If experiments with spoken humor produced negative results, it would be unclear whether the error was based on the speech recognition or the actual humor detection. Thus, this work concentrates on textual humor.

1.3 Motivation for Research in Computational Humor

The usefulness and motivations of computational humor have been discussed in many articles (Raskin & Attardo, 1994; Binsted, 1995; Raskin, 1996; Ritchie, 2001; Mulder & Nijholt, 2002;
Nijholt, 2002; Raskin, 2002; Stock & Strapparava, 2002; Binsted, et al., 2006; Hempelmann, in press).

Binsted (1995) argues that “a limited use of humour within certain facilities – such as the signaling of errors, the reporting of the unavailability of facilities, and certain aspects of information provision – could render a computer system more user-friendly. In particular, such a system should appear less alien, less intimidating, and less patronising.” Lyttle (2001) demonstrated empirically that persuasion is more effective when humor is used. Morkes, Kernal, & Nass (1998) found that users perceive computer systems that employed humor as “more likable and competent” (Hempelmann, in press). Thus, computational humor can increase the friendliness and effectiveness level of human-computer interfaces.

Ritchie (2001) is skeptical about computational humor enhancing human-computer interfaces, but believes that “there is a role for computer modeling in testing symbolic accounts of the structure of humorous texts.” Ritchie also argues that “if we could develop a full and detailed theory of how humour works, it is highly likely that this would yield interesting insights into human behavior and thinking.” Binsted (2006) agrees that “humor provides insight into how humans process language — real, complex, creative language, not just a tractable subset of standard sentences. By modeling humor generation and understanding on computers, we can gain a better picture of how the human brain handles not just humor but language and cognition in general.”

Another motivation for computational humor is human-computer interaction. According to Binsted (2006), “if computers are ever going to communicate naturally and effectively with humans, they must be able to use humor.” Nijholt (2002) suggests that humor is necessary for embodied conversational agents if they are to show personalities.
Education is another area where computational humor could be of benefit. McKay (2002) proposes to use computational humor in second language learning systems. A system for assisting language learners with severe language impairments that uses humor has been proposed and implemented (O'Mara & Waller, 2003; Waller et al., 2005).

It is claimed that a possible use for a sophisticated humor generator is “business world applications (such as advertising, e-commerce, etc...)” (Raskin, 2002). Stock & Strapparava (2006) believe that “an environment for proposing solutions to advertising professionals can be a realistic practical application of computational humor and a useful first attempt at dealing with creative language.” It should be noted that the state of the art of computational humor generators is far from making its stand as a standup comedian.

Raskin (2002) also suggests customer acceptance enhancement and humor detection as valid applications for computational humor. Customer acceptance enhancement “encourages customers to accept an unpopular product by rewarding them with humor [...]”. Information security comes to mind as a prime candidate domain for this endeavor” (Raskin, 2002).

Humor detection is another application and subfield of computational humor. Raskin (2002) suggests that “an application will search for humor, for perversion of the text, for instance, in a presidential address, diplomatic note, or any other deadly serious business. [...]” On the other hand, the same humor detection applications can be used to determine the vulnerable spots in a text to be denigrated, e.g. in a political campaign, and then to work in conjunction with humor generation to create appropriate and effective humor.”

Humor detection can have other applications. It can be used in human-computer interaction, as it might be important to detect if a person is being serious or making a joke. Humor detection can also be used to discard unnecessary or irrelevant information in a web search. A computer
plays the role of an intermediary between a person that needs information and a person that recorded it. If a person makes a query for information about a particular product then the computer must sort through texts to find semantically relevant information. But many texts contain humor. The humor may be irrelevant to the needed information (in which case humor can be ignored), or the needed information may be directly affected by humor in the text (in which case humor needs to be understood). In both cases, humor needs to be detected, and a text should be judged as relevant or irrelevant to the query.

1.4 **Roadmap**

This dissertation is organized as follows: Chapter 2: “Background” introduces information that is relevant to the dissertation from the areas of computer science, linguistics, and psychology. The chapter starts with the “Natural Language” section that discusses what it takes to understand a text, either humorous or non-humorous, from a linguistic point of view, and assesses the difficulties of computational natural language understanding. The “Lexical Access and Semantic and Phonological Associations and Priming” section describes the understanding of the meaning of words, utterances, sentences and texts from a psychological point of view; it takes into account both explicit and implicit information in text, as well as general world knowledge. The “Knowledge Representation” section of Chapter 2 shows how some common knowledge about the world can be represented in computer-readable form, how it can be accessed when needed, and what is involved in accessing and reasoning with it. The chapter ends with a discussion of humor theories that are applied in the project, and gives an overview of computational humor, separated into humor generation and humor detection.
Chapter 3: “Model” provides an overview of the architecture of the humor detector that was used for the investigation of principles of computational humor detection and analysis. It discusses in detail different components, including knowledge acquisition, knowledge representation and algorithms used to classify humorous and non-humorous texts. It also addresses the choices that were made to restrict the domain of humor targeted for the dissertation. The “Semantic Component” section describes the parts responsible for meaning detection of words in the joke: the first part is the knowledge base, containing general information about the world; the second part is a module that finds semantic relationships between words; the third part is a humor recognizer that uses existing theories of humor to determine whether a text, represented in a machine-understandable form, is humorous. The “Orthographic Component” section deals with the spelling of words. This component reads a text from a file, finds numeric word description (frequency of use, familiarity of word) and reports the results. The “Phonological Component” section deals with the pronunciation of words. It contains information about a database of word pronunciations, a table of phoneme similarity, and a module that calculates the pronunciation distance between two similarly sounding words.

Chapter 4: “Experiments” describes the experiments that were conducted on the collections of jokes and non-jokes in the “Experiments” section. The “Sample Size” section explains the decisions that were made about the sample size. The “Joke Selection” section explains the process of joke selection. The “Non-joke Text” section describes how non-jokes were created. The “Joke Recognition: What and Why” section describes the reasons for posing questions addressed in the Experiments section. Finally, the “Other Proposal Questions” section describes
the questions that were asked at the dissertation proposal, and explains why these questions were not addressed in the dissertation.

Chapter 5: “Results” provides a discussion of the experimental results. Each section addresses one of the five questions asked in Chapter 4, namely:

- Is it possible to recognize jokes that are based on lexical ambiguity?
- Is it possible to recognize jokes that are based on the phonological similarity of words?
- Can jokes be recognized by comparing them to already known jokes?
- Can jokes be recognized when an ontology does not have all of the required background knowledge to process the meaning of a text?
- Are some types of jokes easier to recognize than others?

Chapter 6: “Conclusion” summarizes the work and its results, and suggests directions for further research. The chapter is followed by the Bibliography that contains references used in the dissertation, and the Appendices, containing the cost table used for calculating the distance between similarly pronounced words; the high-level algorithm of the joke detector and its use based on an example; and the jokes that were used in the experiments.
2 BACKGROUND

Computational detection of humor is a broad, exciting and challenging field. To narrow the domain, the focus is on humor as expressed in written texts, as opposed to spoken, physical or visual humor. This chapter gives an overview of the literature that is relevant to the computational processing of natural language texts, in general, and humorous texts, in particular. Because humor expressed in texts is based on natural language, it is a natural place to start. To illustrate the difficulties of humor detection, the background section follows example of texts in a language other than English. This also approximates what the computer sees in a text unless provided with a separate semantic component for the language.

Text 2: Штирлиц вышел на поляну и осмотрелся: голубые ели. Штирлиц подошел поближе. Голубые разбежались.

Text 3: Штирлиц посмотрел в окно: голубые ели. Присмотревшись, он понял, что голубые не только ели, но и пили.

Text 4: Штирлиц шел по лесу: голубые ели. Присмотревшись, он понял, что рядом с елями росли березы.

What you see are two Russian jokes and one non-joke. If you don't read/speak/understand Russian, you see strings of characters that don't mean much to you. The next section will demonstrate what it takes to analyse natural language texts in an unfamiliar language.

1 This sentence by itself demonstrates some difficulties of natural language processing: the word “natural” is used in two different meanings.
2.1 Natural Language

For thousands of years people have analyzed everyday uses of human language and the linguistic and social structures that support those uses. A great deal is understood about the structures and functions of languages. Far more remains to be discovered (Finegan, 2004). There are many areas of human language (natural language) that are studied: morphology, phonology, syntax, semantics, pragmatics, etc. Most of these areas are studied for both computational and non-computational purposes. This section will give an overview of the semantic aspect of natural language. Semantics is the study of meaning.

Text understanding requires meaning detection of individual words and utterances that contain these words. The intended meaning of words or utterances may not match their literal meaning. Metaphors, idioms, proverbs or other figures of speech can be used in a text. For example, a literal meaning of the question “do you know what time it is?” is whether one has knowledge of time, while one of the intended meanings of the question is share information about time.

Understanding of the meaning of a text may not come from information in the text by itself: prior world knowledge and willingness to cooperate in understanding of the utterance’s intended meaning may be needed. For example, consider the following text (Winograd & Flores, 1995):

Text 5: Is there water in the fridge?

According to Winograd & Flores, a typical person will think about drinking water, but not about drops of water that may exist on lettuce located in the fridge. If there is no pitcher or bottle of water in the fridge, but there are drops of water on the lettuce, a person expects a

---

2 See Attardo (1994) for an excellent overview.
negative reply, while a computer may answer positively. This happens because Text 5 does not precisely define what kind of water is needed. The literal meaning of Text 5 differs from its intended meaning. However, a human does not need a precise definition of the type of water because most people are not interested in water on lettuce. This shared knowledge may be omitted without causing any problem for humans, but it needs to be present for correct computer results.

A computer needs this knowledge just as people need knowledge about unfamiliar facts or situations. Suppose a person comes to a previously unvisited country with an unfamiliar culture, but is able to understand the language. Things that are obvious to people of that culture have to be explained to the newcomer. A computer can be treated as a newcomer to people’s culture. In order for a computer to understand what something means, the new information has to be explained in terms known to the computer. Now suppose that a person comes in contact with an unfamiliar culture, but is not fluent in the language of the culture. The person knows only some words. This is the position that a computer is in regards to natural language. It may recognize the words, it may access some predefined grammar rules, but understanding the meaning of a sentence is a different task.

In the beginning of this chapter, you saw three texts (Text 2, Text 3, Text 4): two of them are Russian jokes and one is a non-joke. Upon careful examination, you may notice that several strings appear multiple times: Штирлиц and ели appear 4 times, and the word голубые appears 5 times. Unfortunately, knowing the number of occurrences, we still don't know what the words mean and how they are used. Unless there is a meaning associated with each string (or several strings together), there is no hope of detecting if these texts are humorous.
Each word can have several meanings\(^3\), as shown below for Text 2:

- Штирлиц: not available
- вышел: emerge, issue, march out, step down, step out, come out, get out, go out
- на: against, at
- поляну: glade, clearing in the woods
- и: and
- осмотрелся: look about
- голубые: azure, blue, homosexual\(^4\)
- ели: fir, fir-tree, spruce, eat, grub, make, meal, pick, touch, walk into, have
- подошел: get around, befit, do, dovetail, draw, match, near, pertain, rise, step up, suit, walk up, come
- поближе: around, within call, at the fore, on one’s hands, nearabout, near, in the region of, thereabout, in the vicinity of, close up
- разбежались: disband

The list above is similar to what a computer receives when taxonomies, without any interconceptual relationships, are used. In order to correctly choose the senses from the lists, some relationships between pairs of senses must be preserved. For example, “come out” requires a “place” (where the coming out occurs) and an “agent” (who comes out); “look about” requires an agent and a place; “walk up” requires an object to walk to, or a place. Otherwise, one is left

\(^3\) As taken from http://dictionary.paralink.com/.

\(^4\) This sense is not listed in the dictionary.paralink.com.
with a meaningless permutation of all senses, where selecting a sequence at random is as good as any other selection.

Also notice that the word Штирлиц does not have a translation. This is not surprising because the word indicates a name of a double agent in a popular Soviet movie about WWII. The hero was popular enough for the Soviet ruler of the time, Leonid I. Brezhnev, to declare him a Hero of the Soviet Union (or was that a joke too?). The joke cannot be processed completely without any knowledge about this agent. Thus, not only does the computer need to know what the words mean individually and how they can be used together, but it also needs access to the specific knowledge that may be required to understand the joke.

The next section explains how words can be brought together through semantic relationships to understand the meaning of a sentence.

2.1.1 Sentence Meaning and Semantic Roles

Sometimes all required knowledge is supplied in the natural language sentence. To understand the meaning of the sentence, even if all required knowledge is supplied by the text, a computer has to understand relationships between words in the sentence. Consider the following text (Weitzman et al., 2006):

Text 6: What did the paint give the wall on their first anniversary? A new coat.

The first sentence can be split into several parts: paint gives something to the wall, the time of the event is their first anniversary. The second sentence explains that what was given the wall was a new coat. Now, suppose, the text is reworded:

Text 7: What was given to the wall by paint on their first anniversary? A new coat.
The meaning of the text remains the same, while the first sentence structure changed. The meaning of both texts can be described with semantic roles.

The research on semantic roles started with pioneering work by Gruber (1965), Fillmore (1968) and Jackendoff (1972). A thematic role (also called theta role or case role\(^5\)) is “the semantic role played by an argument in relation to its predicate” (Radford, 2004: 250). Different books and articles use different sets of thematic roles. An absence of agreement on a necessary meaning representation set is not surprising, and is similar to an absence of agreement on knowledge representation primitives, described in Section 2.3. The downside of these differences is the difficulty in comparing the performance of the different approaches. The upside is a freedom to choose appropriate, in a researcher’s point of view, resources and roles. To illustrate the differences, three sample lists of thematic roles and their definitions taken from Finegan (2004), Nirenburg & Raskin (2004) and Jurafsky & Martin, (2000) are shown respectively in Table 2-1, Table 2-2 and Table 2-3. In this dissertation, thematic roles from both tables are used.

<table>
<thead>
<tr>
<th>Thematic Role’s Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGENT</td>
<td>The responsible initiator of an action</td>
</tr>
<tr>
<td>PATIENT</td>
<td>Entity that undergoes a certain change of state</td>
</tr>
<tr>
<td>EXPERIENCER</td>
<td>The non-responsible initiator of an action</td>
</tr>
<tr>
<td>INSTRUMENT</td>
<td>The intermediary through which an agent performs the action</td>
</tr>
<tr>
<td>CAUSE</td>
<td>Natural force that brings about a change of state</td>
</tr>
<tr>
<td>RECIPIENT</td>
<td>That which receives a physical object</td>
</tr>
<tr>
<td>BENEFACTIVE</td>
<td>That for which an action is performed</td>
</tr>
<tr>
<td>LOCATIVE</td>
<td>The location of an action or state</td>
</tr>
<tr>
<td>TEMPORAL</td>
<td>The time at which the action or state occurred</td>
</tr>
</tbody>
</table>

Table 2-1: Thematic roles (Finegan, 2004)

\(^5\) This dissertation uses all three names interchangeably.
### Case Role’s Name | Definition
--- | ---
AGENT | The entity that causes or is responsible for an action
THEME | The entity manipulated by an action
PATIENT | The entity that is affected by an action
INSTRUMENT | The object or event that is used in order to carry about an action
SOURCE | A starting point for various types of movement and transfer
DESTINATION | An end point for various types of movement and transfer
LOCATION | The place where an event takes place or where an object exists
PATH | The route along which an entity travels, physically or otherwise
MANNER | The style in which something is done

Table 2-2: Case roles (Nirenburg & Raskin, 2004)

### Thematic Role’s Name | Definition
--- | ---
AGENT | The volitional causer of an event
EXPERIENCER | The experiencer of an event
FORCE | The non-volitional cause of the event
THEME | The participant most directly affected by an event
RESULT | The end product of an event
CONTENT | The proposition or content of a propositional event
INSTRUMENT | An instrument used in an event
BENEFICIARY | The beneficiary of an event
SOURCE | The origin of the object of a transfer event
GOAL | The destination of an object of a transfer event

Table 2-3: Thematic roles (Jurafsky & Martin, 2000)

In Text 6 and Text 7 examples, the verb *give* assigns thematic role *AGENT* to the word *paint*, thematic role *RECIPIENT* to the word *wall* and thematic role *TEMPORAL* to the phrase *their anniversary*. The same thematic role works for the word *wall* of Text 8. It happens because the underlying meaning of the first sentences in these jokes is the same even though different verbs are used.

Text 8: What did the wall receive from the paint on their anniversary? A new coat.

---

6 Assuming that paint can responsibly initiate an action.
While the meaning of the jokes’ first sentence is the same, the words *wall* and *paint* have different syntactic categories. In Text 6 *paint* is the subject and *wall* is the object, while in Text 8 *paint* is the object and *wall* is the subject. While it may be tempting to link the meaning of the words in a sentence to their syntactic category, this small example shows that it can be a dangerous enterprise. Perhaps a better example is Text 9, given by Finegan (2004: 200), where the subject *it* has no semantic or thematic role.

Text 9: It became clear that the government had jailed him there.

Part of Fillmore’s (1968) motivation for thematic roles was the use of them as an intermediary between conceptual structures’ semantic roles and “their more language-specific surface grammatical realization as subject or object” (Jurafsky & Martin, 2000: 609). However, the absence of 1-1 mapping between syntactic categories and thematic roles makes their computational recognition difficult. Additionally, thematic roles cannot show all semantic relationships between words in a sentence. Jurafsky & Martin (2000: 611) explain: “this is because thematic roles are only relevant to determining the grammatical role of NP and PP arguments, and play no part in the realization of other arguments of verbs and other predicates.” Furthermore, Jurafsky & Martin (2000: 611-612) argue that “thematic roles are only useful in mapping the arguments of verbs, but nouns, for example, have arguments as well.”

There are many possible approaches for solving the problem of the thematic roles. Most approaches rely on defining additional semantic roles that address arguments not covered by thematic roles. Assigning semantic roles to words or utterances is generally referred to as semantic role labeling.

7 See Jurafsky & Martin (2000) for an excellent overview.
Semantic role labeling can be done manually, automatically, or semi-automatically. The manual approaches result in very high accuracy, but are time demanding. The accuracy for manual approach could be defined as the percentage of cases when the agreement between multiple researchers is reached. Most automatic approaches involve statistical semantic role labeling. Gildea & Jurafsky (2002) use FrameNet\(^8\) knowledge base (Ruppenhofer et al., 2006) to automatically label semantic roles. The authors report 82% accuracy in semantic role identification of pre-segmented constituents. When the system segments continuants by itself and then identifies semantic roles, 65% precision and 61% recall are reported.

Pighin & Moschitti (2007) use a tree kernel based shallow semantic parser for semantic roles extraction. The authors separate the task into a thematic role boundary detection subtask and a role classification subtask. The results of overall system performance are: precision value = 81.58% and recall value = 70.16%. Haghighi, Toutanova, & Manning (2005,2007) also use a shallow parser for semantic role labeling. They report 79.54% precision value and 77.39% recall value.

Pradhan, Ward, & Martin (2007) varied corpora for parser training, semantic role training and tests. Their results show that when the same corpus is used for parser training, semantic role training and semantic role testing, the precision and recall values are similar to those reported in articles above (precision value ranges from 77.4% to 77.5%, recall value ranges from 62.1% to 69.7%). However, when a corpus used for training differs from the testing corpus, especially for semantic labeling task, the result are significantly lower: 63.7% precision value and 55.1% recall value.

---

\(^8\) See Section 2.3 for a more detailed information about FrameNet.
A semi-automatic ontological and lexical acquisition technology is introduced in the Ontological Semantics theory (Nirenburg & Raskin, 2004). The meaning of natural language texts is automatically derived through a compositional process, taking into an account language-dependent lexicon and language-independent ontology (see also Section 2.3.6 for ontology discussion). Each entry in the lexicon contains syntactic and semantic structures. The syntactic structure shows possible uses of the word in relation with other constituents that are shown as variables in the syntactic structure. The semantic structure defines and interrelates all variables of the syntactic structure through the concepts and semantic roles defined in the ontology. The semantic and syntactic structures are manually created for each entry. A separate entry is created for each meaning of each word. The lexicon and the ontology are a product of many man-years.

Let us go back to Text 2 (Штирлиц вышел на поляну и осмотрелся: голубые ели. Штирлиц подошел поближе. Голубые разбежались.) If semantic roles are used, the first sentence can be analyzed as follows:

- Вышел is a verb or event that has an AGENT Штирлиц and DESTINATION на поляну.
- Осмотрелся is a verb or event that has an AGENT Штирлиц and THEME голубые ели.

The problem is that голубые ели can mean two different things:

- Object ели with COLOR голубые.
- Event ели with AGENT голубые.

This brings us to the issue of ambiguity in natural language.

### 2.1.2 Ambiguity in Natural Language

Let’s look at an example in English, which is easier for an English speaker to understand. Suppose all semantic roles are correctly identified for the first sentence of Text 6 (What did the
paint give the wall on their first anniversary? A new coat): the paint is the agent (or the giver), the wall is the recipient (or given to), and their first anniversary is temporal. But what does the word wall mean? What does the word coat mean in the phrase a coat of paint?

Many words have multiple meanings. The word wall, according to the American Heritage First Dictionary (2006), has two meanings:

- A wall is a side of the room. Most rooms have a ceiling, a floor, and four walls.
- A wall is something that is built to keep one place apart from another. Example: The farmer built a stone wall around his field.

According to WordNet (Fellbaum, 1998), the word wall has eight different meanings as a noun:

- An architectural partition with a height and length greater than its thickness; used to divide or enclose an area or to support another structure
- Anything that suggests a wall in structure or function or effect
- A layer (a lining or membrane) that encloses a structure
- A difficult or awkward situation
- A vertical (or almost vertical) smooth rock face (as of a cave or mountain)
- A layer of material that encloses space
- A masonry fence (as around an estate or garden)
- An embankment built around a space for defensive purposes

Making a decision as to which possible meanings of the word are used is referred to as word-sense disambiguation. Word sense disambiguation is a major area in natural language processing. Usually, the word can be disambiguated in its context. Text 6 is an example of lexical ambiguity: the context is insufficient to determine the meaning of the word wall. Notice that lexical ambiguity does not always result in humor: the ambiguity of the word wall is not what creates the joke. However, it can be a “humor enabler” (Oaks, 1994). This usually happens when a listener expects one meaning of a word, but is forced to use another meaning:

Text 10: Teacher: Mrs. Jones, I asked you to come in to discuss Johnny’s appearance.
Mrs. Jones: Why? What’s wrong with his appearance?
Teacher: He hasn’t made one in this classroom since September.

Text 10 enabled two meanings of the word appearance:
• Visible aspect of a person or thing

• The act of appearing in public view

The reader expects the first meaning of the word, but the first meaning does not work in the context of the third sentence. Thus, the reader is forced into the second meaning of the word. Both meanings are triggered by the context of their corresponding sentences. Other meanings of the word *appearance* are ignored by a human.

Syntactic ambiguity is an ambiguity form produced by more than one possible syntactic interpretation of a sentence. Syntactic ambiguity by itself is not a sufficient condition for humor production, but similar to the lexical ambiguity, it can be a humor enabler:

Text 11: Question: What has four wheels and flies?
Answer: A garbage truck

The ambiguity is created by two different parse trees of the question part of the joke, as shown in Figure 2-1.

Figure 2-1: Different parse trees for Text 11.

The listener first interprets the question according to the parse tree on the left, but the answer forces the interpretation corresponding to the parse tree on the right. These two parse trees are not the only possible ones: the interpretation of “four (wheels and flies)” is syntactically possible, but does not make sense semantically.
Another form of ambiguity is semantic. Semantic ambiguity arises when a common expression has a number of possible interpretations. As an example, consider:

Text 12: How do you turn a regular scientist into a mad scientist? Step on her toes.

The phrase “step on her toes” can have meanings ranging from literal interpretation of the words to idiomatic expressions.

Referential ambiguity arises when it is unclear to what an expression refers. Pronouns often enable referential ambiguity. As an example, consider (Tinholt & Nijholt, 2007):

Text 13: Did you know that the cops arrested the demonstrators because they were violent?

The pronoun they can be referred to demonstrators or cops. The use of referential ambiguity for computational humor generation was investigated in Tinholt & Nijholt (2007).

Another source of misunderstanding in speech comes from words that sound alike or similar. Many children’s jokes and puns use similar sounding words to create jokes:

Text 14: Boy: Why didn’t you pull a rabbit out of your hat? Magician: Because I just washed my hare.

The joke is based on identical pronunciation between the words hare and hair (ˈhɛər) and two possible meanings of the sentence “I just washed my ˈhɛər”. The first meaning of the sentence is “I just washed my rabbit”; the second meaning of the sentence is “I just washed the top of my head”.

2.1.3 Detecting Humor and Statistical Language Processing

Statistical language processing can be very powerful for answering some questions, especially when the knowledge required to answer those question is not explicitly available. Answering
questions such as “What are the common patterns that occur in the language?” (Manning & Schuetze, 2002: 4) should be done with statistical processing. Answering questions similar to “What are the frequent meanings of this word in this domain?” can be done from semantically-tagged corpora. It may even be possible to answer such question as “What is the exact meaning of this word?” if it is possible to fully disambiguate the word in its context. The problems start when we ask questions like “What is the exact meaning of this sentence?”

Manning & Schuetze (2002: 3) state that “it is just not possible to provide an exact and complete characterizations of well-formed utterances that cleanly divides them from all other sequences of words, which are regarded as ill-formed utterances.” The systems that are unable to accurately conclude if a sentence is valid should not be expected to conclude that something is humorous. The conclusion is not an impossible task all the time: if a book of jokes is entered into the system and each joke is tagged as such, it may be possible to conclude that an identical or highly-similar text, occurring elsewhere, is also a joke. But the task of a fully accurate humor detection using non-semantic techniques is likely to be impossible. A non-semantic humor detection line of reasoning is similar to the reasoning in Text 15:

**Text 15:** What are the chances of meeting a dinosaur on the street? 50/50. You either meet him or you don’t.

Computational humor detection and recognition cannot be solved by answering the question “What are the frequent patterns that occur in humor?” This question has a precondition: it assumes that the texts used to answer the questions are indeed humorous. These same patterns occur in texts that are not humorous or that are not perceived as humorous: the sentence “A man walks into a restaurant.” is not humorous in itself! Humor is not a matter of frequency, it is a matter of semantics.
Hempelmann & Raskin (2008) state:

Semantics can be done semantically or with the method *du jour* in order to avoid having to do it semantically. These methods, largely statistical and formal-logical, tend to hide the lacking motivation for their application to an issue that requires access to meaning beyond neat formalism. [...] Practically, doing semantics semantically means to emulate human processing. This entails to acquire massive human-like knowledge resources, in particular a language independent ontology (conceptual hierarchy) and a language specific lexicon (anchored in the ontology). [...] If you don’t want to do semantics semantically, for reasons briefly speculated about below, you would usually use syntactic/statistical/tagging/annotating methods in order to not have to acquire any semantic resources. In other words, your aim is to guess meaning from non-meaning phenomena, like co-occurrence and other surface structure properties of language.

This dissertation takes the semantic approach to humor in text. The evidence of non-semantic techniques used for semantic role selection has been demonstrated by the Cognitive Computation Group’s semantic role labeling tool with Text 6, Text 8, discussed in the Model section.

As to the Russian texts examples, the first joke, Text 2, can be translated as follows, using the list of senses provided on page 13: “Schtirlitz came out to the clearing in the woods and looked around: blue spruces (alternative meaning: homosexuals ate). Schtirlitz stepped closer. Homosexuals disbanded.” The joke works because the combination голубые ели can mean either “blue spruce”, in which case it is an adjective and a noun, or a “homosexuals ate”, in which case it is a noun and a verb. The last sentence, however, requires голубые to be a noun, thus rejecting a more common choice of an adjective in the first sentence.

The joke in Text 3 also uses the same combination: голубые ели. It can roughly be translated as: “Schtirlitz looked at the window: голубые ели. Upon a more careful examination, he realized that the homosexuals not only ate, but drank too.” While this joke uses the same word pair as an ambiguity enabler (Oaks, 1994), the body of the joke is different. A non-semantic approach would hardly catch it. What makes the non-semantic approach to joke detection difficult in this
case is the common use of the “blue spruce” utterance, indicating a type of an evergreen. A non-

joke is demonstrated in Text 4: “Schtirlitz walked in the woods: голубые ели. Upon a more
careful examination, he realized that next to the spruces grew birches.”

Similarly to Texts 2-4, the coat of paint joke in Text 6 (Section 2.1.1) would not be
recognized by the non-semantic approaches. For further discussion, see Section 3.1.2.

2.2 **Lexical Access and Semantic and Phonological Associations and Priming**

When a text is read by a human, descriptions of some words or situations are activated. This

section is not concerned with the computational aspect of text processing, but rather with how
human process words and sentences that they read, and what associations arise from reading
words. While the humor detector does not exactly simulate how people recognize humor, it takes
some of it into the account.

The section discusses empirical results of word detection, when the detection is manipulated
by showing other words, related or non-related to the word in question. These results shed light
on when and why the needed meanings of ambiguous words are activated, as well as why non-
existing words or words with non-fitting meanings can activate the needed meaning to process
the text.

For a person to understand a text, the text has to be relevant to this person’s world knowledge
and coherent. World knowledge is our mental model; it is the information about the world that
we have accumulated; it is “a cognitive structure that represents some aspect of our
environment” (Carroll, 2004: 231). For a text to be understood, all information does not have to

25
be present in the text itself; it can be activated by our mental model, or world knowledge. For example, consider Text 16 (Carroll, 2004: 213):

Text 16: John bought a cake at the bake shop. The birthday card was signed by all of the employees. The party went on until after midnight.

The sentences are not explicitly connected with information, yet we seem to make sense out of them, and make them into one story. In order to connect the sentences in Text 16 into a coherent story, schemata\(^9\) have to be activated. The schemata are triggered by words in the text. The words have to have semantic relation between them. In Text 16, John probably bought a cake with some purpose in mind. The next sentence mentions a birthday card. Birthday cards have some purpose. A cake together with a birthday card activates the purpose of celebrating somebody’s birthday. The celebration can be in the form of a party, thus connecting the third sentence.

Text understanding depends on activation of the appropriate schemata. If the schemata are not activated, the individual utterances make sense, but the text as a whole can be unclear. Consider the following text as an example (Dooling & Lachman, 1971: 218):

With hocked gems financing him, our hero bravely defied all scornful laughter that tried to prevent his scheme. “Your eyes deceive,” he had said, “an egg not a table correctly typifies this unexplored planet.” Now three sturdy sisters sought proof, forging along sometimes through calm vastness, yet more often over turbulent peaks and valleys. Days became weeks as many doubters spread fearful rumors about the edge. At last from nowhere welcome winged creatures appeared signifying momentous success.

\(^9\) A schema (plural: schemata) is a mental structure that specifies and represents information about some aspect of the world.
It may be difficult to comprehend the above text without determining and activating the proper schema. The task is easier when one knows that the text is about Christopher Columbus discovering America. Information about Columbus discovering America activates the proper schema. The schema represents relevant background knowledge and makes comprehension possible.

Kozminsky (1977) found that people recall the same texts with different appropriate titles differently. Kozminsky suggests that “titles can alter the comprehension of a text by affecting the selection of information from a text and the organization of this information in memory.” According to Carroll (2004), information that is central to the schema that is in effect during comprehension is well remembered, while other details are misplaced.

Schemata activation plays an important role in humorous texts comprehension as well. Suppose people have a conversation about athletic organizations for kids. It is likely that a schema about athletic clubs has been activated. When these people hear the question “Do you believe in clubs for young people?” they are likely to think about clubs as organizations since this is the schema that is commonly activated. Suppose the conversation was not about athletic organizations, but about child abuse. Upon hearing the same sentence, it is possible that a different schema is activated. It could have information about hitting children with some objects.

Now, let’s look at Text 17:

Text 17: Do you believe in clubs for boys?
Only when kindness fails.

It could be argued that if the conversation was about child abuse, there is a chance that the “Athletic organization” schema was not activated. If the schema of hitting children is activated first, there may not be a need for the athletic organization schema activation. Moreover, even if
the athletic organization schema is activated, there may not be any surprise (which leads to humor in this joke). Thus, for computational detection of humor, it is crucial to activate the correct schema.

Schemata can be activated by specific words. People store and process words in mental lexicons. The lexicons contain lexical entries with information about a word’s spelling, pronunciation, meaning and syntactic category. All of it, to some extent, contributes to schemata activation.

According to Yates, Locker, & Simpson (2004), “models of visual word perception must attempt to account for the effects of words’ orthographic, phonological, and semantic characteristics”. Since reading text requires visual word perception, all three components are considered for a lexical entry activation.

In order to simulate schema activation, it is important to understand whether all words equally contribute to it, or some have more weight in the task. It turns out that the process of accessing lexical information from the memory depends on a number of factors. One of the factors is word frequency.

2.2.1 Word Frequency

The role of word frequency in lexical access\textsuperscript{10} has been demonstrated in many studies. Foss (1969) asked participants to comprehend a passage with low- and high-frequency words and listen to a particular phoneme at the same time. The results indicate that the time for phoneme monitoring was higher when the phoneme followed a low-frequency word, compared to a high-frequency word.

\textsuperscript{10} Lexical access means word retrieval from our mental lexicon.
A similar effect is shown by lexical decision tasks. A lexical decision task involves showing words and pseudowords to a participant, who decides whether the displayed string of letters is a real word. The results of the lexical decision tasks (Gardner, Rothkopf, & Lafferty, 1987; Balota & Chumbley, 1984) show that the response time is higher for low-frequency words than for high-frequency words.

Rayner & Duffy (1986) showed that eye fixation on low frequency words is longer than high-frequency words in normal reading task. In the results, the authors’ concluded that low-frequency words are accessed slower than high-frequency words.

2.2.2 Lexical Ambiguity

The word frequency experiments raise a question. How to calculate word frequency if a word has a single meaning is clear. What happens if a word is polysemous (i.e., can have multiple meanings)? This form of ambiguity is referred to as lexical ambiguity. Lexical ambiguity is important to computational humor detection as many humorous texts are based on lexical ambiguity.

According to Pexman & Lupker (1999), the existence of the polysemy effect is controversial, as some articles seem to contradict each other (cf. Borowsky & Masson, 1996; Clark, 1973; Foster & Bednall, 1976; Gernsbacher, 1984; Hino & Lupker, 1996; Jastrzembski, 1981; Kellas, Ferraro, & Simpson, 1988; Rubenstein, Garfield, & Millikan, 1970). Some studies showed that the response time of the lexical decision task was lower for polysemous words, while others did not. Thus, what affects lexical decision in humor processing is unclear.

Studies of eye movements show that the fixation times of ambiguous words depend on frequencies of meanings (Duffy, Morris, & Rayner, 1988; Rayner & Frazier, 1989). According
to these studies, fixation times are longer when meanings of the word have similar frequencies, and shorter when one of the meanings is significantly more frequent. Moreover, there is no difference in fixation time between unambiguous words and ambiguous words with one frequent meaning.

2.2.3 Orthographic Effects

Lexical access can also be affected by existence of words that are spelled alike. Words with a one-letter-difference in spelling are called orthographic neighbors of the target word; a collection of these words is called an orthographic neighborhood. According to Andrews (1997), most studies (Andrews, 1992; Sears, Hino, & Lupker, 1995) have shown a decrease of reaction time for a high orthographic neighborhood density when the words are mixed.

A word’s orthographic neighbors can have higher or lower frequencies than the target word. Grainger & Jacobs (1996) found that a target word’s lexical access in Dutch and French is delayed when its neighbors have a higher relative frequency. Carreiras, Perea, & Grainger (1997) demonstrated the same effect in Spanish. Perea & Pollatsek (1998) found the same effect in English. Furthermore, the same effect was found in normal reading in English (Pollatsek, Perea, & Binger, 1999).

Acha & Perea (2008) investigated the role of a word’s transposed-letter\(^{11}\) neighbors in normal silent reading, and found the same effect. Acha & Perea (2008) conclude that their findings and others’ (Perea & Pollatsek, 1998; Pollatsek, Perea, & Binger, 1999; Bowers, Davis, & Hanley, 2005; Davis & Lupker, 2006) “indicate that there are effects of lexical competition

\(^{11}\) Two letters are swapped, for example, causal : casual.
when reading text for meaning. Furthermore, lexical competition occurs not only for substitution, addition or deletion neighbors, but also for transposed-letter neighbors.”

2.2.4 Phonological Effects

There is evidence that phonology plays a role in lexical access. Jared, McRae, & Seidenberg (1990) found that words that have only one pronunciation are named faster than words with multiple pronunciations. Ziegler, Montant, and Jacobs (1997) reported a similar effect on lexical decision tasks.

Pexman, Lupker, & Jared (2001) found that homophones\textsuperscript{12} result in a slower response on lexical decision tasks. Van Orden (1987) found that homophones in semantic categorization tasks affect the speed of the response as well.

Yates, Locker, & Simpson (2004) demonstrated that phonological neighborhood density affects lexical decisions. The authors argue that “[t]his finding has important implications for models of visual word recognition, which must include the influence of phonology on the perception of written letter strings. These results are consistent with a fully interconnected model comprising orthographic, phonological, and semantic components.”

Phonological priming occurs when a word presented earlier activates another, phonologically related word. Experiments with phonological priming (Humphreys, Evett, & Taylor, 1982) showed that words that sound similar to the source are activated faster.

A number of eye-movement studies (Inhoff & Topolski, 1994; Rayner, Pollatsek, & Binder, 1998; Sparrow & Miellet, 2002) show that phonological codes are activated early during eye

\textsuperscript{12} These are words that are pronounced the same.
fixation. These results also point to the influence of phonology on the perception of written words.

### 2.2.5 Semantic Effects

Finally, there is evidence that semantic priming\(^\text{13}\) affects lexical decision tasks (Meyer & Schvaneveld, 1971). Semantic priming occurs when a word presented earlier activates another, semantically related, word. The results showed that words that are semantically related to a source are activated faster.

Experiments with semantic neighborhoods also showed that words with a large semantic neighborhood are responded to faster than words with small semantic neighborhoods (Buchanan, Westbury, & Burgess, 2001). The semantic neighborhood is defined as the number of words associated with the target word.

These results show that the speed of recognition of a target word depends on the number of words related to the target. While it is a nice result, in practice, it is unclear how the number of related words can be calculated.

To summarize the results of this section, the speed of word recognition depends on many factors, including: the number of meanings the word has, the number of pronunciations the word has, the number of words with similar spelling, the number of words semantically associated with the word in question, and word frequency. All of these factors should be considered in a humor detection task.

\(^{13}\) Semantic priming can be defined as the increased sensitivity to a stimulus due to prior experience or information.
2.3 **Knowledge Representation**

General information about the world should be represented in a computer-readable and meaningful form in order for the computer to be able to use it. For a computer to process the first Schtitz joke (Text 2 above), the meanings of the words in the text must be presented also in a computer-readable format. For example, it needs to know that

- Штирлиц is a *person* (and what *person* means).
- Осмотрелся is a *voluntary event* and it has an *agent*, which is a *person*, and a *theme*, which is an *object* or *event*.
- Ели can be a *tree*, and a *tree* has *branches*, a *stem*, a *root*, and *needles*; and the *needles* have some *color*.

The field that is concerned with representing information about the world is called knowledge representation. There are many different ways to computationally represent information. This section gives a brief overview of the some approaches.

According to Sowa (2000), knowledge representation is a multidisciplinary subject that developed as a branch of artificial intelligence and applied theories and techniques from logic, ontology and computation fields. Sowa describes these three fields as follows:

- Logic provides the formal structure and rules of inference.
- Ontology defines the kinds of things that exist in the application domain.
- Computation supports the applications that distinguish knowledge representation from pure philosophy.

“Knowledge representation is the application of logic and ontology to the task of constructing computable models for some domain” (Sowa, 2000: xii). The information discussed in this section will mostly relate to the fields of logic and ontology.
Mylopoulos & Levesque (1984) provide a different categorization of representational schemes (Hayes, 1974) from Sowa: logical representational schemes, procedural representational schemes, network representational schemes, and structured representational schemes. This classification originated from declarative and procedural schemes (Winograd, 1975). Declarative schemes are further classified into logical and network schemes (Mylopoulos, 1981).

Network schemes are further extended into structured schemes (Luger & Stubblefield, 1998: 294). Each of the four categories below has its strengths and weaknesses:

• Logical representation schemes use expressions in formal logic to represent a knowledge base. Inference rules and proof procedures apply this knowledge to problem instance. First order predicate calculus is the most widely used logical representation scheme. […]
• Procedural representation schemes represent knowledge as a set of instructions for solving a problem. This contrasts with declarative representations provided by logic and semantic networks. […] A production system may be seen as an example of a procedural representation scheme.
• Network representation schemes capture knowledge as a graph in which the nodes represent objects or concepts in the problem domain and the arcs represent relations or associations between them. Examples of network representations include semantic networks, conceptual dependencies, and conceptual graphs.
• Structured representation schemes. Structured representation languages extend networks by allowing each node to be a complex data structure consisting of names slots with attached values. These values may be simple numeric or symbolic data, pointers to other frames, or even procedures for performing a particular task. Examples of structured representations include scripts, frames, and objects.

This section will discuss logical, network and structured representation schemes. The procedural schemes will not be touched as irrelevant to the body of this dissertation.

2.3.1 Semantic networks

A semantic network is a network representation scheme. It presents knowledge as a graph, with the nodes corresponding to concepts or facts and arcs corresponding to relationships or associations between the nodes (Luger & Stubblefield, 1998). According to Collins & Loftus (1975), semantics nets were invented by Richard Richens in 1956 as “interlingua” for machine
translation of natural languages. They were developed by Robert Simmons in the early 1960s; and later featured in semantic memory models (Quillian, 1967). The semantic networks have property arcs between concepts and IS-A arcs, that introduce hierarchical and instance relationships. An example of semantic network for *frog* is shown in Figure 2-2.

![Semantic Network Example](image)

Figure 2-2: An example of semantic network (Baader, 1999).

According to Luger & Stubblefield (1998: 303), graph relationship notation has little advantage over predicate calculus; “the power of network representation comes from the definition of links and associated inference rules that define a specific inference such as inheritance”.

### 2.3.2 Conceptual graphs

Conceptual graphs (Sowa, 1984) are systems that are based on semantic networks and existential graphs. Conceptual graphs “provide means of representing case relations, generalized quantifiers, indexicals, and other aspects of natural language” (Sowa, 2000). The nodes of a conceptual graph are either concepts or conceptual relations. An example of conceptual graph is shown in Figure 2-3. The graph corresponds to a sentence “John is going to Boston by bus”.

---

35
Conceptual graphs (CGs) are concerned with knowledge representation as well as reasoning (Sattler, Calvanese, & Molitor, 2003). Reasoning with conceptual graphs includes deciding where one graph is subsumed by another, as well as whether a graph is valid. Conceptual graphs can be translated to predicate calculus (Sowa, 2000).

![Conceptual Graph Example](image)

Figure 2-3: An example of conceptual graph (Sowa, 2000)

There is a trade-off between expressivity of a language and difficulty of reasoning (see for instance, Brachman and Levesque 1984): the more expressive the language is, the more difficult is reasoning. “Since conceptual graphs can express all of first-order predicate logic (Sowa, 1984), these reasoning problems are undecidable for general CGs” (Sattler, Calvanese, & Molitor, 2003). Reasoning problems in simple conceptual graphs, on the other hand, are decidable. Luger & Stubblefield (1998) note that while conceptual graphs can be represented in a predicate calculus syntax, they support some special-purpose inference mechanisms that are not part of predicate calculus, such as join and restrict.

### 2.3.3 Frames

Frame systems were introduced by Minsky (1975), as an alternative to logic-representation schemes. A frame represents well-known stereotypical situations in a static data structure. Luger & Stubblefield (1998) present the following frame description by Minsky (1975):

<table>
<thead>
<tr>
<th>Person: John</th>
<th>Agnt</th>
<th>Go</th>
<th>Dest</th>
<th>City: Boston</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
Here is the essence of the frame theory: When one encounters a new situation (or makes a substantial change in one’s view of a problem) one selects from memory a structure called a “frame.” This is a remembered framework to be adapted to fit reality by changing details as necessary.

Each frame contains slots. The slots can represent various information, such as: frame identification, the relationship of this frame to other frames, descriptors of requirements for frame matching, procedural information on use of the structure described, frame default information, and new instance information (Luger & Stubblefield, 1998). An example of frames can be seen in Figure 2-4.

![Figure 2-4: Example of a frame (Luger & Stubblefield, 1998)](image)

Frames can be seen as extensions to semantic networks. In semantic networks “properties are restricted to primitive, atomic ones, whereas, in general, properties in frame systems can be
complex concepts described by frames” (Sattler, Calvanese, & Molitor, 2003). Frames make it easier to hierarchically organize knowledge (Luger & Stubblefield, 1998). The procedural information is an important feature of frames, compared to semantic networks because some knowledge does not adapt well to declarative representation (Luger & Stubblefield, 1998).

While frames were designed as an alternative to logic representation schemes, they were criticized by Hayes (1977, 1979) on the basis that the declarative part of the frames could be captured using first-order predicate logic (Baader, 1999). According to Sattler, Calvanese, & Molitor (2003), no precise semantics could be given to the non-declarative part of frames. They conclude that the expressive power of frames and the quality of their reasoning algorithms cannot be compared to other knowledge representation formalisms.

There are many applications that use frames for natural language processing. FrameNet (Ruppenhofer et al., 2006) is one of them. FrameNet is a lexicon-building effort, which attempts to select words with particular meanings and describes the frames that underline these meanings. Each frame corresponds to a predicate, with possible arguments that it can take. Some frames in FrameNet can be seen in Figure 2-5.

<table>
<thead>
<tr>
<th>Frame</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame(TRANSPORTATION)</td>
<td></td>
</tr>
<tr>
<td>frame_elements(MOVER(S), MEANS, PATH)</td>
<td>scene(MOVER(S), move along PATH by MEANS)</td>
</tr>
<tr>
<td>frame(DRIVING)</td>
<td>inherit(TRANSPORTATION)</td>
</tr>
<tr>
<td>frame_element(DRIVER (=MOVER), VEHICLE (=MEANS), RIDER(S) (=MOVER(S)), CARGO (=MOVER(S)))</td>
<td>scenes(DRIVER starts VEHICLE, DRIVER controls VEHICLE, DRIVER stops VEHICLE)</td>
</tr>
</tbody>
</table>
FrameNet is an interesting application because it is used in semantic role labelers (as discussed in Sentence Meaning and Semantic Roles section).

### 2.3.4 Conceptual Dependencies and Scripts

Some events happen in particular order. A script (Schank & Abelson, 1977) is a knowledge representation formalism that describes a stereotypical sequence of events. The script was designed to organize conceptual dependency (Luger & Stubblefield, 1998).

An attempt to model the deep semantic structure of natural language texts was made in Schank and Rieger’s (1974) conceptual dependency theory. Conceptual dependencies have conceptual categories and primitives on which the meaning representation is built (Luger & Stubblefield, 1998: 305):

- ACTs: actions
- PPs: objects (picture producers)
- AAs: modifiers of actions (action aiders)
- Pas: modifiers of objects (picture aiders)

The ACTs can be represented in terms of the following components:

- ATRANS: transfer a relationship (give)
- PTRANS: transfer physical location of an object (go)
- PROPEL: apply physical force to an object (push)
- MOVE: move body part by owner (kick)
- GRASP: grab an object by an actor (grasp)
- INGEST: ingest an object by an animal (eat)
• EXPEL: expel from an animal’s body (cry)
• MTRANS: transfer mental information (tell)
• MBUILD: mentally make new information (decide)
• CONC: conceptualize or think about an idea (think)
• SPEAK: produce sound (say)
• ATTEND: focus sense organ (listen)

The primitives are used to define conceptual relationships that can describe associations of concepts or semantic roles. Conceptualizations also include tense and mode information.

An example of a script, described in terms of conceptual dependencies is shown in Figure 2-6 and Figure 2-7.

<table>
<thead>
<tr>
<th>Scene 1: Entering</th>
<th>Scene 2: Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>S PTRANS S into restaurant</td>
<td>(Menu on table) (W brings menu) (S asks for menu)</td>
</tr>
<tr>
<td>S ATTEND eyes to tables</td>
<td>S MTRANS signal to W</td>
</tr>
<tr>
<td>S MBUILD where to sit</td>
<td>W PTRANS W to table</td>
</tr>
<tr>
<td>S PTRANS S to table</td>
<td>S MTRANS ‘need menu’ to W</td>
</tr>
<tr>
<td>S MOVE S to sitting position</td>
<td>W PTRANS W to menu</td>
</tr>
</tbody>
</table>

Figure 2-6: Restaurant script, scenes Enter and Order (Luger & Stubblefield, 1998)
Scene 3: Eating

C ATRANS F to W
W ATRANS F to S
S INGEST F

(Option: Return to Scene 2 to order more; otherwise, go to Scene 4)

Scene 4: Exiting

W MOVE (write check)
W PTRANS W to S
W ATRANS check to S
S ATRANS tip to S
S PTRANS S to M
S ATRANS money to M
S PTRANS S to out of restaurant

Figure 2-7: Restaurant script, scenes Eating and Exiting (Luger & Stubblefield, 1998)

Each script, as shown in Figure 2-8, has several components: an entry condition, results, props, roles, and scenes. The entry conditions have to be true for the script to be initialized. The results show what is true after the script terminates. The props are objects or things that are usually involved in a typical situation described by the script. The roles show the participants of the event. The scenes, shown in Figure 2-6 and Figure 2-7, are sequences of the event.

Script: Restaurant

| Track: Coffee Shop | Entry cond: S hungry
| Props: Tables Menu F=Food Check Money | S has money
| Roles S=Customer W=Waiter C=Cook M=Cashier O=Owner | Results: S has less money
| | O has more money
| | S is not hungry
| | S is pleased (optional)

Figure 2-8: Components of a Restaurant script (Luger & Stubblefield, 1998)

The advantages of the conceptual dependencies (and scripts) are that they reduce ambiguity by providing a formal theory of natural language semantics; and it provides “a canonical form for the meaning of sentences” (Luger & Stubblefield, 1998: 308). However, Woods (1985)
pointed out that it is questionable whether natural language system can be reduced to a canonical form. Additionally, the primitives are inadequate to capture all aspects of natural language.

2.3.4.1 Scripts in Ontological Semantics

This section is largely based on Raskin et al. (2003), the first attempt to incorporate scripts into Ontological Semantics theory.

In theoretical linguistics, script-based theories of semantics were proposed by Fillmore (1985) and Raskin (1986). Scripts are typical sequences of events that occur in standard, easily identifiable situations. In a state visit, for instance, the agents may include the principals such as heads of state, their assistants, body guards, police and the press. All these agents and their props will fill the theme roles and other properties in the sub-events of the standard event sequence for a state visit. “The component events are often optional; alternatively, some component events stand in a disjunctive relation with some others (that is, of several components only one may actually be realized in a particular instantiation of the overall complex event), and their relative temporal ordering may be fuzzy” (Raskin et al. 2003).

If a sentence in a text is recognized as belonging to a certain script, the processing of the subsequent sentences, including disambiguation, can be considerably facilitated “by the expectation that propositions contained in them are instantiations of event types that are listed as components of the activated script” (Raskin et al. 2003). The expected difficulties include: the recognition of a sentence as initiating a script or belonging to it; the possible deviation of some sentences and clauses from the script; the variability of the grain size in the sentences. Of great importance to the scripts is the identification of the goals of all agents and of the plans that are used to fulfill these goals.
2.3.5 Description Logics

Description logics (DLs) are logical reconstructions of frame-based knowledge representation languages, that are used to provide well established declarative semantics to capture the meaning of the most popular features of structured representations of knowledge.

The semantics of the language is provided via the interpretation function that maps symbolic definitions to real world objects. An example of syntax and semantics of some of the constructors is provided in Figure 2-9.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Example</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic concept</td>
<td>A</td>
<td>Human</td>
<td>$A' \subseteq A'$</td>
</tr>
<tr>
<td>atomic role</td>
<td>R</td>
<td>likes</td>
<td>$R' \subseteq A' \times A'$</td>
</tr>
<tr>
<td>conjunction</td>
<td>C \sqcap D</td>
<td>Human \sqcap Male</td>
<td>$C' \cap D'$</td>
</tr>
<tr>
<td>disjunction</td>
<td>C \sqcup D</td>
<td>Nice \sqcup Rich</td>
<td>$C' \cup D'$</td>
</tr>
<tr>
<td>negation</td>
<td>¬C</td>
<td>¬Meat</td>
<td>$\Delta' \setminus C'$</td>
</tr>
<tr>
<td>existential restriction</td>
<td>\exists R.C</td>
<td>\exists hasChild.Human</td>
<td>${x \mid \exists y. &lt; x, y &gt; \in R' \wedge y \in C'}$</td>
</tr>
<tr>
<td>value restriction</td>
<td>\forall R.C</td>
<td>\forall hasChild.Blond</td>
<td>${x \mid \forall y. &lt; x, y &gt; \in R' \Rightarrow y \in C'}$</td>
</tr>
</tbody>
</table>

Figure 2-9: Syntax and semantics of some constructors (Horrocks, 2005)

Knowledge is represented as a tuple of three sets of axioms - terminological (concept) inclusions (TBox), role inclusions (RBox) and assertions of objects memberships to concepts and roles (ABox). An example of a terminological box is given in Figure 2-10, an example of an assertion box is given in Figure 2-11.

\[
\begin{align*}
\text{Woman} & \equiv \text{Person} \sqcap \text{Female} \\
\text{Man} & \equiv \text{Person} \sqcap \neg \text{Woman} \\
\text{Mother} & \equiv \text{Woman} \sqcap \exists \text{hasChild.Person} \\
\text{Father} & \equiv \text{Man} \sqcap \exists \text{hasChild.Person} \\
\text{Parent} & \equiv \text{Father} \sqcup \text{Mother} \\
\text{Grandmother} & \equiv \text{Mother} \sqcap \exists \text{hasChild. Mother} \\
\text{MotherWithoutDaughter} & \equiv \text{Mother} \sqcap \forall \text{hasChild.} \neg \text{Woman} \\
\text{Wife} & \equiv \text{Woman} \sqcap \exists \text{hasHusband.Man}
\end{align*}
\]

Figure 2-10: TBox example (Baader & Nutt, 2003)
Different classes of description logics offer different expressivity. Since expressivity increases reasoning difficulty, it is important to find a class of DLs that is expressive enough and still decidable for reasoning problems. Figure 2-12 shows different components of expressivity of description logics. A description logic class can be formed by combining the “symbols” from the right column of Figure 2-12.

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Semantics</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top</td>
<td>$T$</td>
<td>$\Delta'$</td>
<td>$\mathcal{AL}$</td>
</tr>
<tr>
<td>Bottom</td>
<td>$\bot$</td>
<td>0</td>
<td>$\mathcal{AL}$</td>
</tr>
<tr>
<td>Intersection</td>
<td>$C \sqcap D$</td>
<td>$C' \cap D'$</td>
<td>$\mathcal{AL}$</td>
</tr>
<tr>
<td>Union</td>
<td>$C \sqcup D$</td>
<td>$C' \cup D'$</td>
<td>$\mathcal{AL}$</td>
</tr>
<tr>
<td>Negation</td>
<td>$\neg C$</td>
<td>$\Delta' \setminus C'$</td>
<td>$\mathcal{C}$</td>
</tr>
<tr>
<td>Value restriction</td>
<td>$\forall R.C$</td>
<td>${a \in \Delta' \mid \forall b.(a,b) \in R' \rightarrow b \in C' }$</td>
<td>$\mathcal{AL}$</td>
</tr>
<tr>
<td>Existential quant.</td>
<td>$\exists R.C$</td>
<td>${a \in \Delta' \mid \exists b.(a,b) \in R' \land b \in C' }$</td>
<td>$\mathcal{E}$</td>
</tr>
<tr>
<td>Unqualified number restriction</td>
<td>$\geq nR$</td>
<td>${a \in \Delta' \mid \exists b.(a,b) \in R' \land b \in C' } \models \geq n$</td>
<td>$\mathcal{N}$</td>
</tr>
<tr>
<td>Unqualified number restriction</td>
<td>$\leq nR$</td>
<td>${a \in \Delta' \mid \exists b.(a,b) \in R' \land b \in C' } \models \leq n$</td>
<td>$\mathcal{N}$</td>
</tr>
<tr>
<td>Unqualified number restriction</td>
<td>$= nR$</td>
<td>${a \in \Delta' \mid \exists b.(a,b) \in R' \land b \in C' } \models = n$</td>
<td>$\mathcal{N}$</td>
</tr>
<tr>
<td>Qualified number restriction</td>
<td>$\geq n.R.C$</td>
<td>${a \in \Delta' \mid \exists b.(a,b) \in R' \land b \in C' } \models \geq n$</td>
<td>$\mathcal{N}$</td>
</tr>
<tr>
<td>Qualified number restriction</td>
<td>$\leq n.R.C$</td>
<td>${a \in \Delta' \mid \exists b.(a,b) \in R' \land b \in C' } \models \leq n$</td>
<td>$\mathcal{N}$</td>
</tr>
<tr>
<td>Qualified number restriction</td>
<td>$= n.R.C$</td>
<td>${a \in \Delta' \mid \exists b.(a,b) \in R' \land b \in C' } \models = n$</td>
<td>$\mathcal{N}$</td>
</tr>
<tr>
<td>Role-value-map</td>
<td>$R \subseteq S$</td>
<td>${a \in \Delta' \mid \forall b.(a,b) \in R' \rightarrow (a,b) \in S' }$</td>
<td>$\mathcal{E}$</td>
</tr>
<tr>
<td>Role-value-map</td>
<td>$R = S$</td>
<td>${a \in \Delta' \mid \forall b.(a,b) \in R' \leftrightarrow (a,b) \in S' }$</td>
<td>$\mathcal{E}$</td>
</tr>
<tr>
<td>Agreement and disagreement</td>
<td>$u_1 = u_2$</td>
<td>${a \in \Delta' \mid \exists b \in \Delta'.u_1'(a) = b = u_2'(b)}$</td>
<td>$\mathcal{F}$</td>
</tr>
<tr>
<td>Agreement and disagreement</td>
<td>$u_1 \neq u_2$</td>
<td>${a \in \Delta' \mid \exists b_1,b_2 \in \Delta'.u_1'(a) = b_1 \neq b_2 = u_2'(b)}$</td>
<td>$\mathcal{F}$</td>
</tr>
<tr>
<td>Nominal</td>
<td>$I'$</td>
<td>$I' \subseteq \Delta'$ with $\models I' \models = 1$</td>
<td>$\mathcal{O}$</td>
</tr>
</tbody>
</table>
Description logics have been used in natural language processing (Weischedel, 1989; Allgayer et al., 1989; Fehrer, et al., 1994; Herzog & Rollinger, 1991; Stock, et al., 1991; Samek-Lodovici & Strapparava, 1990; Lavelli, Magnini, & Strapparava, 1992; Franconi, 1994; Wahlster, 2000; Rychtyckyj, 1999). While even most expressive description logics may not be expressive enough to represent all aspects of natural language, they offer reasoning capabilities that are needed by a humor detector.

2.3.6 Ontologies

An ontology is a “specification of a conceptualization” (Gruber, 1993). Often, it is a hierarchical data structure containing all the relevant entities, their relationships and rules within a domain, and provides a shared understanding about a domain of interest (Sowa, 2000: 493):

An ontology is a catalog of the types of things that are assumed to exist in a domain of interest $D$ from the perspective of a person who uses a language $L$ for the purpose of talking about $D$. The types in the ontology represent the predicates, word senses, or concept and relation types of the language $L$ when used to discuss topics in the domain $D$.

Some entities in an ontology, called primitives, cannot be defined in terms of other entities. The primitives are “linked” directly to the top node of the ontology. The primitives used depend on a knowledge engineer, responsible for their construction. Sowa (2000) defines seven primitives: Independent, Relative, Mediating, Physical, Abstract; Continuant, Occurrent. Primitives may change with ontology refinement. Non-primitive entities are generated by combining the primitives or other already derived entities. An example of a top-level ontology is shown in Figure 2-13.

Non-primitive entities are generated by combining the primitives or other already derived entities. For example, Form is generated by combining Independent and Abstract; Script is
generated by combining *Form* and *Occurrent*. Notice that the script concept in this ontology is not the same as the scripts of conceptual dependencies. General concepts are formed in the upper ontology, which contains entities that are domain independent.

![Top level ontology](Sowa, 2000)

Ontologies can range from formal to informal. “An informal ontology may be specified by a catalogue of types that are either undefined or defined only by statements in a natural language. A formal ontology is specified by a collection of names for concept and relation types organized in a partial ordering by the type-subtype relation” (Sowa, 1999). Informal ontologies are glossaries and dictionaries, thesauri and taxonomies. Formal ontologies are frames, general logic ontologies, and description logic ontologies.
There is a disagreement between researchers as to whether glossaries, taxonomies, and other types of informal ontologies can actually be considered ontologies. In this dissertation, the term “ontology” is used interchangeably with the term “formal ontology” meaning those that contain information about concepts and relationships between them.

An ontology can be represented in any knowledge representation formalism. The choice of formalism depends on the interest domain and preferences of a knowledge engineer. This dissertation will use description logics to represent an ontology.

2.4 Humor Theories

Work on humor theories has a long history, and, to this day, the true nature of humor is still being debated. There is no universally accepted theory of humor that explains “what is funny, why it is funny, how it is funny, when it is funny, and to whom it is funny” (Raskin, 1985: 5).

Most humor models and theories can be divided into the three classes. According to the most common classification (Morreall, 1983; Raskin, 1985; Ritchie, 2004), humor theories can be based on: incongruity, relief/release or hostility/superiority/aggression/malice/disparagement.

Attardo (1994) divides humor theories into three similar classes, but gives the classes different names, as shown in Figure 2-14.

<table>
<thead>
<tr>
<th>Cognitive</th>
<th>Social</th>
<th>Psychoanalytical</th>
</tr>
</thead>
<tbody>
<tr>
<td>incongruity</td>
<td>hostility</td>
<td>release</td>
</tr>
<tr>
<td>contrast</td>
<td>aggression</td>
<td>sublimation</td>
</tr>
<tr>
<td></td>
<td>superiority</td>
<td>liberation</td>
</tr>
<tr>
<td></td>
<td>triumph</td>
<td>economy</td>
</tr>
<tr>
<td></td>
<td>derision</td>
<td></td>
</tr>
<tr>
<td></td>
<td>disparagement</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2-14: Types of Humor theories (Attardo, 1994: 47)
Incongruity-based theories suggest that humor arises from something that violates an expectation. Many supporters of incongruity in humor have emphasized the importance of surprise in a joke (Raskin, 1985).

Superiority theories are based on the observation that people laugh at other people’s infirmities, especially if they are enemies (Suls, 1976). This class of theories of humor goes back to the philosophers of Ancient Greece, who maintained that people laugh at the misfortunes of others for joy that they do not share them (Raskin, 1985; Attardo, 1994).

Release/relief theories explain the link between humor and laughter. The principle for release-based theory is that laughter “provides relief for mental, nervous and psychic energy, and this ensures homeostasis after a struggle, tension, and strain” (Raskin, 1985: 38). “The most influential proponent of the release theory is certainly Freud” (Attardo, 1994: 50).

The classes can be summarized (Ritchie, 2000): “…cognitive/incongruity approaches concentrate on the humorous stimulus, social/hostility approaches consider the interpersonal effects, and psychoanalytical/relief proposals emphasise the audience’s reaction.”

In this work, the concentration is on the stimulus aspect of humor, as per Ritchie’s classification above. Thus, more attention is paid to the cognitive or incongruity based class. There is a debate as to whether incongruity alone is sufficient for laughter. Some researchers argue that incongruity is the first step of a multistage process and that a retrieval of information resulting in satisfactory resolution of incongruity is a necessary step for a humorous response (Suls, 1976; Ritchie, 1999). This theory is called Incongruity-Resolution (IR) theory, as it requires an extra step: resolution of the incongruity. The incongruity itself is the necessary condition for humor, while resolution of incongruity is the sufficient condition (Hempelmann, 2003).
2.4.1 Script-based Semantic Theory of Humor

Script-Based Semantic Theory of Humor (SSTH) (Raskin, 1985) is a linguistic theory that is used in this work to determine if a text is humorous. The theory is neutral with respect to the three classes.

According to the theory, there are two conditions for a text to be humorous:

- A text has to be compatible, fully or in part, with two different scripts.
- The two scripts with which the text is compatible are opposite, and with which the text must overlap fully or partially.

The compatibility of the text with two scripts is the necessary condition for humor; the oppositeness of the scripts is the sufficient condition (Raskin, 1985; Hempelmann, 2003). A script “is an enriched, structured chunk of semantic information, associated with word meaning and evoked by specific words. The script is a cognitive structure internalized by the native speaker and it represents the native speaker’s knowledge of a small part of the world. […] What is labeled ‘script’ here has been called ‘schema’, ‘frame’, ‘daemon’, etc. […] Formally or technically, every script is a graph with lexical nodes and semantic links between the nodes” (Raskin, 1985: 81). Scripts were further developed in Ontological Semantics (Nirenburg and Raskin, 2004) and are described in the Scripts section (Section 2.3.4).

The scripts can be linguistic, general knowledge, restricted, or individual, as shown in Figure 2-15. Linguistic scripts are known to any “average,” “standard” native speaker (adult, reasonably educated, mainstream culture, etc). General knowledge scripts, such as crossing the street or going to a store, are known to a large number of people and are not affected by their use of language. Restricted knowledge scripts are known to a smaller number of people and are not
affected by their use of language. Good examples of restricted knowledge scripts and individual scripts can be found in Attardo (1994: 247).

![Figure 2-15: Script arrangement (Raskin, 1985)](image)

In Text 1 (“A man walks into a bar. Ouch.”), there are at least two associated scripts. The first one describes entering a bar. It is evoked by the words *man, walk in, and bar*. The second one describes hitting something. It is evoked by the words *man, walk in, bar, and ouch*. The overlap is obvious: words are activated by both scripts. The oppositeness between the two scripts is *pleasure vs. pain*\(^{14}\). These two scripts are not the only ones that might be associated

---

\(^{14}\) As it was pointed out by C.F. Hempelmann (2007), personal communication.
with the joke. Other activations may result from “man walking into a bar” being the setup of another joke; and, unexpectedly, being a punchline\textsuperscript{15}, triggered by the word “ouch”.

According to Raskin, intentional verbal humor is based on ambiguity that is created deliberately. However, ambiguity by itself is not enough for humor: the scripts must not only be opposed, they must be so unexpectedly. This means that if you expect both scripts to appear, as you start reading “man walks into…” and you do not find the oppositeness surprising, you will not find Text 1 humorous. Some other examples of oppositeness are: \textit{good : bad, real : unreal, money : no money, life : death}, etc (Raskin, 1985).

Unfortunately, in computational terms, it is unclear when the scripts overlap, when they oppose; and, at what level of abstraction the scripts should be placed. A detailed algorithmic representation may not have been necessary at the time the theory was published, but it is needed for computational humor analysis.

Script overlap can be treated as an intersection of sets (Attardo, Hempelmann, & Di Maio, 2002), when scripts are looked at as sets, as shown in Figure 2-16. Sets A and B overlap in AB; C and D show script opposition. According to Attardo, Hempelmann, & Di Maio (2002) “two overlapping scripts (A and B) are opposed when within the complementary sets of the intersection we can locate two subsets (C and D) such that the member(s) of the subset C are the (local) antonyms of the member(s) of the subset D”

\textsuperscript{15} SO setup:punchline was suggested by C.F. Hempelmann (2007), personal communication, as well.
Figure 2-16: Script overlap and opposition (Attardo, Hempelmann, & Di Maio, 2002)

Hempelmann (2003: 21) proposes to treat oppositeness as “situational, contextual, or local antonyms.” This may make sense to a person, but it is difficult to create an algorithm to recognize situational or contextual antonyms. Additionally, there is no complete listing of scripts that can overlap and oppose, which makes computational humorous text recognition difficult.

2.4.2 General Theory of Verbal Humor

The General Theory of Verbal Humor (Attardo & Raskin, 1991) is a linguistic theory of humor that is built upon the notion of script overlap and script oppositeness. The theory describes jokes in terms of six knowledge resources (Attardo, 1994):

- **Script Opposition (SO)**: deals with script overlap and oppositeness presented in Script-based Semantic Theory of Humor (SSTH).
- **Logical Mechanism (LM)**: accounts for the way in which the two scripts in the joke are brought together.
- **Situation (SI)**: the “props” of the joke, the textual materials evoked by the scripts of the joke that are not necessarily funny.
- **Target (TA)**: any individual or group from whom humorous behavior is expected.
- **Narrative Strategy (NS)**: the rhetorical structure of the text; that is the “genre” of the joke, such as the riddle, 1-2-3 structure, question and answer, etc.
• **Language (LA):** the actual lexical, syntactic, phonological, etc., choices at the linguistic level that instantiate all the other choices; language is responsible for the position of the punchline.

According to the General Theory of Verbal Humor, each joke can be viewed as a 6-element vector, specifying the instantiation of each parameter (Ruch, Attardo, & Raskin, 1993):

\[
\text{Joke: } \{\text{LA, SI, NS, TA, LM, SO}\}
\]

The only element that is optional in a joke description is the **Target (TA)**.

This theory makes it possible to compare jokes using the 6-element vector. Two jokes are different if at least one parameter of the above six is different in the two jokes. The difference value between two jokes is dependent on the number of different parameters.

The GTVH, just like the SSTH, does not provide formal definition of its parameters. The LM parameter is further discussed and formalized to a large degree in Attardo, Hempelmann, & Di Maio (2002), which also provides a taxonomy of LMs.

It could be argued that some of the parameters in the joke vector are derivatives of other parameters. Even so, the parameters in the joke vector representation make a convenient joke comparison possible, providing a useful shortcut in the cumbersome calculations of the primitive features of humorous text.

Consider the following joke:

Text 18: A computer walks into a bar. Ouch.

This joke, as Text 1, can activate several scripts, resulting in different SO’s: **setup : punchline, real : unreal, pain : pleasure.** Depending on what SO is activated for Text 1 and Text , their similarity would be different. For example, suppose Text 18 activates the **real : unreal**
SO. This SO will not work for Text 1, as there is nothing unreal about the situation(s) described. As long as Text 1 activates SO other than real : unreal, the two are perceived as having more differences than if both jokes activated the setup : punchline SO.

2.4.3 Puns

The domain in this work is narrowed further by considering short humorous texts, where script oppositeness is triggered by:

- (P): phonological similarity of two words that belong to opposing scripts,
- (A): a word that has two different meanings, each meaning belonging to two different scripts.

The texts in the (P) category are often referred to as puns: they are either perfect puns, when two words or utterances sound alike, or imperfect puns, when there is a slight difference in pronunciation. Imperfect, or paronomasic, puns were described by Hempelmann (2003). Hempelmann’s analysis can be applied to perfect puns as well. In this work, the texts of the (A) category are treated as perfect puns and commonly referred to as puns.

Hempelmann distinguishes puns and simple wordplay, as shown in Table 2-14. Both a pun and wordplay have the same logical mechanism: cratylism (Hempelmann, 2003):

This illogical but powerful reasoning can be summarized in the following syllogism: If meaning motivates sound, and sound is identical (similar), then meaning must be identical (similar). This line of paralinguistic reasoning is named after Kratylos, a participant in the eponymous Platonic dialogue, who argues for the natural, motivated, non-arbitrary relationship between sound and meaning (cf. Attardo 1994: 152).

The difference between a joke, a wordplay and a non-joke, as shown in Table 2-4, is made by the presence or absence of SO, described by the General Theory of Verbal Humor. The table
also shows that if a *SO* is present but the *LM* is not cratylism, then the joke does not depend on the wordplay.

<table>
<thead>
<tr>
<th>Semantic Oppositeness SO</th>
<th>Cratylism LM</th>
<th>Interpretation</th>
<th>Common name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Present</td>
<td>Present</td>
<td>Punning joke</td>
<td>Pun</td>
</tr>
<tr>
<td>Absent</td>
<td>Present</td>
<td>wordplay</td>
<td>Pun</td>
</tr>
<tr>
<td>Present</td>
<td>Absent</td>
<td>(A) non-punning joke</td>
<td>(B) non-joke ambiguity</td>
</tr>
<tr>
<td>Absent</td>
<td>Absent</td>
<td>Non-joke text</td>
<td></td>
</tr>
</tbody>
</table>

Table 2-4: Pun vs. Wordplay in terms of SO and LM (Hempelmann, 2003)

The following examples, provided in Hempelmann (2003), for interpretations listed in Table 2-4 are:

1. Labia majora: the curly gates.
   curly -> pearly
2. Magnet: To some, it is what you find in a bad apple.
   magnet -> maggot
3a. Gobi Desert Canoe Club
   (non-punning *LM*: direct juxtaposition)
3b: The square circle ate five freedoms
4. In case of an emergency, pull cord

As an example of a text that can be a joke or a non-joke, consider Text 18:

Text 18: Knock Knock.
   Who’s there?
   Cantaloupe.
   Cantaloupe who?
   Can’t elope tonight – Dad’s got the car.

Hempelmann argues that Text 18 is a joke only if *SO* is found, and proposes that one possible *SO* is *food : sex*. Unless this *SO*, or any other *SO* is found, Text 18 is not a joke, but merely wordplay. If the found *SO* is too feeble, then Text 18 will still be identified as a joke, but probably not good enough to be enjoyed.
Following this reasoning, if puns are viewed as a mathematical set, and wordplays are also viewed as a set, then the set of puns is a subset of the set of wordplays. The jokes that are used in this dissertation are both puns and non-puns. The texts that contain wordplay, but where SO could not be found will not be considered as jokes, and will not be reported as such. They are reported as texts containing wordplay.

If an adult fails to detect a valid opposition in a Knock Knock joke, it has to be classified as a non-joke for that adult. Why then do children typically find such a joke funny? It is unlikely that a young child will recognize the sex : nonsex SO in Text 18, but that child may unconsciously sense a different SO. This ‘default’ SO may include a violation of something normal for a child, perhaps, even a speech pattern, thus resulting in the normal : abnormal SO.

2.4.4 N+V Theory of Humor

Does everything that contains SO and LM result in humor? According to Veatch (1998), it may not. Veatch does not look only at verbally expressed humor. Instead he provides the following conditions for all types of humor:

V: The perceiver has in mind a view of the situation as constituting a violation of some affective commitment of the perceiver to the way something in the situation ought to be. That is, a “subjective moral principle” of the perceiver is violated.

N: The perceiver has in mind a predominating view of the situation as being normal.

Simultaneity: The N and V understandings are present in the mind of the perceiver at the same instant in time.

The conditions are individually necessary, and jointly sufficient for humor. A “subjective moral principle” is defined as “the way things should be.” Veatch explains N + V in terms of
commitment of a perceiver to a principle that is being violated in a potentially humorous situation. When there is no violation (and no commitment to a violated principle) the perceiver will not see the violation, will not get offended by it, and will not see humor in it. When there is a violation, but it is perceived as normal, the commitment to this violation is weak. The perceiver will therefore see the violation, will not be offended by it, and will see humor in it. When there is a violation and it is perceived as not normal, the commitment to this violation is strong. The perceiver will see the violation, will be offended by the violation, and will not find it humorous. The theory can be looked at through interpersonal effects humor-theory-class glasses.

While applying this theory would not add another layer of difficulty to computational detection of humor (at least until computers have feelings and emotions), its use should be considered in computational humor generation. It is not enough to be able to generate a potentially successful joke, a computer would have to know its audience when (and if) the joke is delivered. A computer would have to know “subjective moral principles” of the person with whom it communicates.

### 2.5 Computational Humor

The motivation for research in computational humor was presented in Section 1.3. This section reviews previous efforts in this area.

Binsted & Ritchie (1996) propose separating computational humor approaches into top-down and bottom-up approaches. The top-down approaches provide general humor methods that should be applicable to all components. The bottom-up approaches “concentrate on a methodology for starting to develop [the theory of humor] together with a preliminary proposal for some of the concepts which might go to make up the theory” (Ritchie, 2004).
Most computational humor approaches are bottom-up approaches in that they select a restricted subset of humor for generation or recognition.

Several humor generators and very few computational humor recognizers have been attempted. The small number of the attempts is partly due to the difficulty in accessing a context sensitive, computationally based world model.

Several computational humor systems have been proposed in languages other than English (Yokogawa, 2002; Binsted & Takizawa, 1998). This section will not describe them, as the systems are largely language-dependent, and the intricacy of the specific languages would have to be described in order to understand the systems.

### 2.5.1 Existing Computational Humor Generators

Most humor generators that exist today are (humor) theory - free. Because this dissertation is concerned with textually expressed humor, only systems that rely on textual humor are discussed. Additionally, most computational humor generators use templates to generate jokes. The templates contain enough information that their syntactic structure does not have to be computationally verified – it is a given. The humor generators do not generate sentences, but rather fill in the blanks with appropriate words. They are hardly generators in the sense of natural language generators. They do, however, fill in the blanks in a way that sometimes results in humor. While these generators provide interesting results, they are not necessarily of high quality.

One of the first humor generators, LIBJOG (Raskin & Attardo, 1994) provided a template to associate a specified target group with a stereotypical trait and provided an appropriate situation for the classic light-bulb-changing joke:
Template: How many [lexical entry head] does it take to change a light bulb? [number\_1]. [number\_1 – number\_2] to [activity\_1] and [number\_2] to [activity\_2].

To some extent, the system uses the General Theory of Verbal Humor (GTVH) for its underlying structure. According to Hempelmann (in press), the authors were aware of the program’s zero intelligence: “each such lexicon entry is already a ready-made joke” (Raskin, 1996). “The main thrust of LIBJOG was to expose the inadequacy of such systems and to emphasize the need to integrate fully formalized large-scale knowledge resources in a scalable model of computational humor” (Hempelmann, in press).

JAPE-1 (Binsted & Ritchie, 1994), JAPE-2 (Binsted, 1996) and JAPE-3 were also template based, but used a humor-independent lexicon. Instead of using one template to generate riddles, it used several. Ritchie (2003) describes JAPE as follows:

JAPE’s rules were of four different types, performing different tasks in the overall model: schemata (defining configurations of lexemes underlying riddles), sentence forms (patterns of fixed text with slots for further text strings to be inserted), templates (defining conditions for particular items to be inserted into sentence forms), and SAD generation rules (which create abstract linguistic structures from lexemes). The term “SAD”, originally short for small adequate description, is used here for compatibility with Binsted’s terminology. JAPE-3 contains 15 schemata, 18 SAD rules, 9 templates and 9 sentence forms.

An example of JAPE’s sentence form is below:

cross (L1, L2. Str) -->
What do you get when you cross np(L1) and np(L2)? Str.

Text 19 is an example of a riddle generated by JAPE, using the above sentence form:

Text 19: What do you get when you cross a murderer with breakfast food? A cereal killer.
An extension to JAPE is the STANDUP project (Ritchie et al., 2006). STANDUP is a software that helps a child to “explore sounds and meanings by making up jokes, with computer assistance” (Ritchie et al., 2007). STANDUP adapted the joke construction methods from JAPE. It is the (first) practical application for computational humor.

Lessard & Levison (1992) created a model for a particular type of linguistic humor, Tom Swifties. “Tom Swifties are pun-like utterances ascribed to the character Tom, in which a manner adverb enters into a formal and semantic relation with the other elements in the sentence.” Text 20 shows an example of a Tom Swifty.

Text 20: “I hate seafood,” said Tom crabbily

Again, everything produced by this generator is in the form of a template:

Template: “SENTENCE” said Tom ADV[manner]

The adverb in the Template must have a phonetic link to the meaning of at least one word in the SENTENCE, and be semantically related to it.

The Mnemonic Sentence Generator (McDonough, 2001) is a program that directly addresses human-computer interaction. The program converts an alphanumeric password into a humorous sentence, thus making it easier for a user to remember the password. The program addresses an important problem: passwords have become an integral part of everyday life; and good passwords, consisting of both alphabet and numeric characters, are difficult to remember. This system is also template based. It takes an eight character alphanumeric string and transforms it into a sentence, consisting of two clauses connected by the word while. The template for the program is shown below:
The program is loosely based on the Script-based Semantic Theory of Humor (SSTH) as it combines opposite scripts by using a positive verb in the first clause and a negative verb in the second clause. An example of what the program can do is the following sentence, generated from password “AjQA3Jtv”: “Arafat joined Quayle’s Ant, while TARAR\textsuperscript{16} Jeopardized thurmond’s vase” (McDonough, 2001).

HAHA\textsuperscript{14}Acronym (Stock & Strapparava, 2002; Stock & Strapparava, 2005) is another system loosely based on SSTH. “One of the purposes of the project is to show that using standard resources […], and suitable linguistic theories of humor […], it is possible to implement a working prototype” (Stock & Strapparava, 2002). The main tool used is an incongruity detector/generator. It uses WordNet domains such as Medicine or Linguistics and antonymy relations between the domains to create humorous interpretations of existing acronyms. The project inputs an existing acronym and after comparing the actual meaning and context comes up with its humorous parody:

MIT (Massachusetts Institute of Technology) \textasciitilde Mythical Institute of Theology
ACM (Association for Computing Machinery) \textasciitilde Association for Confusing Machinery

According to Hempelmann (in press), WordNet has been augmented for the project to the degree that its own contributions become marginal.

Witty Idiomatic Sentence Creation Revealing Ambiguity In Context (McKay, 2002) is a pun generator that focuses on witticisms based around idioms. This program produces puns and

\textsuperscript{16} Both TARAR and 3 start with the letter “t.”
explanations for the created puns, making it possible for the program to be used as an aid for teaching English idioms to non-native speakers. The puns are produced in three different linguistic forms: question-answer, single sentence and two-sentence sequence.

The program consists of three modules (McKay, 2002):

- Joke Constructor – the module that contains information about elements of a joke. This module uses dictionary of idioms, dictionary of professions, general dictionary, and lexicon.
- Surface-form Generator – the module that uses grammar to convert an input from Joke Constructor into a joke.
- Explanation Generator – this module takes the elements provided by Joke Constructor and uses grammar to generate an explanation of relations between the elements.

An example of a WISCRAIC joke and explanation is Text 21:

Text 21: Who broke the woman’s heart? The cruel deer-keeper.

Ritchie notes that the program “operated by finding sets of items which were related in specific ways, then slotting them into stock textual forms” (Ritchie, 2005).

Hempelmann (2003) proposes “The Ynperfect Pun Selector,” as a complement to a general pun generator based on the General Theory of Verbal Humor. YPS would use heterophonic puns: puns that use a similar sound sequence. It would take any English word as its input and generate a set of words similar in sound, ordered by their phonological similarity that can be used as possible puns for a given target. This generator is based on “a formalized model for the complex phenomenon of punning and heterophonic punning, at all levels of linguistic and humor-theoretical relevance” (Hempelmann, in press). This output could then be entered into a general pun generator for evaluation of the semantic possibilities of the choices produced by YPS. Hempelmann (2003) explains:
For example, ‘dime’ to denote not just a 10¢ coin [daym] but paradigmatically also the meaning of [dæm] as in the slogan ‘Public transportation: It’s a dime good deal.’ YPS’s purpose here is to generate a range of phonologically possible puns given a target word, for example, how we could use not only dam (‘barrier across waterway’) as a homophonic pun to target damn, but also the heterophonic candidates dime (as in the example above), but also damn, dome, dumb, damp, tame, etc.”

The system captures the phonological component of punning through phoneme pairs.

Most of the joke generators described in this section follow one or several predetermined sentence structures. Additionally, the generator does not have to understand the meaning of an entire sentence, it only needs to understand a preselected part of a sentence and then generate a humorous addition to it. Computational generators of humor can also restrict their lexicon or usable background and operational knowledge of the world. Using a small number of words that are humor independent allows a system to claim “humor independent lexicon” and leaves no hope to scale the system from toy examples to larger applications.

These shortcuts are unlikely to work with humor detectors. First, it is very difficult to narrow the sentence structure of existing jokes to several formats (even if the narrative structure of the joke is fixed). Second, the lexicon remains large and world knowledge cannot be restricted to several (dozens) schemata. These reasons alone make it clear why there are not many humor detectors in existence.

2.5.2 Existing Computational Humor Detectors

The Knock Knock joke recognizer (Taylor & Mazlack, 2004) was one of the first bottom-up approaches to humor recognition. It uses letter substitution for wordplay generation, and statistical natural language techniques (N-Grams) for wordplay and punchline validation. The
Knock Knock recognizer was successful at wordplay generation and validation, and recognition of non-jokes. It was less successful in punchline validation. The program was later enhanced to use phoneme substitution for wordplay generation; and parsing for punchline validation.

The One-Liner recognizer (Mihalcea & Strapparava, 2005; Mihalcea & Strapparava, 2006) is another bottom-up approach to humor recognition. The authors define a one-liner as “a short sentence with comic effects and an interesting linguistic structure: simple syntax, deliberate use of rhetoric devices (e.g. alliteration, rhyme), and frequent use of creative language constructions meant to attract the readers attention.” This program classifies sentences into humorous or non-humorous using “(a) heuristics based on humor-specific stylistic features (alliteration, antonymy, slang); (b) content-based features, within a learning framework formulated as a typical text classification task; and (c) combined stylistic and content-based features, integrated in a stacked machine learning framework.”

A similar approach to one-liners was taken by Mihalcea & Pulman (2007). The authors investigated humorous news articles from The Onion and non-humorous news articles from The Los Angeles Times, the Foreign Broadcast Information Service, and the British National Corpus. Naïve Bayes and Support Vector Machine algorithms were used for text classification. The classification accuracy using Support Vector Machines reached 96.8%. The results indicated that there are several features present in humorous texts: human-centric vocabulary, negation, negative orientation, professional communities, and human “weakness”. While the numbers look terrific, it seems to the author of this dissertation that a suicide note, according to the “humorous” features, would also be classified as humorous.

It is this author’s opinion that perhaps it is more fruitful to look for a text meaning, not features, at least as far as humor is concerned.
3 Model

The task of the humor detector is to recognize humor as humans do. Humorous texts are perceived as such if we find them coherent and at the same time incongruous, according to our mental model of the world. It is that mental model that provides the scripts recognized by the Script-based Semantic Theory of Humor (SSTH) or the General Theory of Verbal Humor (GTVH). The mental model also provides the sense or notion of violation and normality for the N+V theory of humor. The resolution of incongruity has to correspond to our mental model as well. The mental model itself results from general world knowledge and personal experiences.

The humor detector consists of three components: orthographic, phonological and semantic. These components correspond to human levels of processing: what we read in a text, what we hear, what we make out of it or how we interpret it. For the humor detector, this can also mean that higher frequency words evoke the scripts. The word frequency values are taken from Francis & Kucera (1967).

The orthographic component (what we read) serves as input to the humor detector. It is bidirectionally connected to the semantic component (how we interpret it). It is important to establish this connection for an attempt to model human language processing. Because we usually try to understand what we read in terms of some semantic structure, there is an orthographic-to-semantic component connection. Ideally, the orthographic component of any language - not just English - would be connected to the same semantic component. In other words, whether a person reads the English word “dog”, the Russian word “собака”, or the Spanish word “perro”, the said word would be connected to a concept in the semantic
component, corresponding to the descriptive definition of a dog. While this dissertation is restricted to English texts, the model can be extended to the use of other languages. At the same time, people are able to express thoughts and their meanings in writing. Therefore, there is the reverse connection (from the semantic component to the orthographic component) or semantic-to-orthographic connection.

The orthographic component is also connected to the phonological\textsuperscript{17} component (how words are pronounced). This connection, also bidirectional, is established for several reasons. The first reason is that one of the goals of this dissertation is to recognize humor that is based on similar sounding words. Because the texts are \textit{read} by a computer, there should be a computationally accessible mapping between the spelling of the words and their pronunciation. The second reason is that, according to some children's and adolescent literacy views, (mis)spelling is influenced by pronunciation (Read, 1986; Treiman, 1985; Moats, 1996), thus justifying the need for a connection from the pronunciation (or phonological) component to the spelling component. There is also evidence that phonological codes, often incorrect (e.g., hypercorrection), are generated from printed words (Ferrand & Grainger, 1996; Frost, 1998; Lukatela & Turvey, 1994; Perfetti & Bell, 1991; Ziegler et al., 2000), thus justifying the connection from the spelling component to the pronunciation (or phonological) component.

The phonological component and the semantic component are also connected bidirectionally. The phonological-to-semantic component connection is obvious: people recognize the meaning of an utterance when the utterance is pronounced. Also, people are able to express what they

\textsuperscript{17} The term is used in its commonly loose linguistic sense as including phonetics.
mean verbally and vocally. Therefore, a connection from the semantic component to the phonological component is justified.

Figure 3-1: Humor detector architecture

The overall architecture of the humor detector is shown in Figure 3-1. The orthographic component is colored blue, the phonological component is colored green, and the semantic component is colored red. The cylinder (rounded tops and bottoms) shapes correspond to information contained in the databases or knowledge bases. As with most knowledge bases, existing information can be retrieved and new information can be entered. Each cuboid shape corresponds to an algorithm underlying the programming code that has a specific purpose. Each

\[^{18}\text{For black and white reproduction, the orthographic component is on the left, the phonological component is on the right, and the semantic component is in the middle.}\]
uses information from the knowledge bases to generate its results. The shapes resembling a paper sheet with a part torn off correspond to small text documents that either serve as input or output, or contain small-size data structures used for computational purposes.

The architecture and functions of the individual components are described in the corresponding subsections.

3.1 **Semantic Component**

The semantic component is the heart and brain of the humor detector. The semantic component consists of three parts, listed in the order they are described. The first part is the knowledge base, which contains general knowledge about the domain as well as specific knowledge described in the texts. The second part is the semantic role labeler, which uses information in the knowledge base to assign semantic relationships to words in the texts, translate the relationships to a machine-readable meaning representation of a text, and insert the specific knowledge contained in the text into the ontology. The third part is the humor analyzer, based primarily on the Script-based Semantic Theory of Humor (SSTH), that determines if a machine-readable meaning representation of the input text results in humor.

3.1.1 **Knowledge Base**

Knowledge can be represented with an ontology. An ontology is a knowledge base that contains concepts and relationships between them that describe a particular domain. An overview of different types of ontologies can be found in Section 2.3.6. The chosen domain for this work is the knowledge available to young children.

The original ontology was built from knowledge manually extracted from *The American Heritage First Dictionary* (2006). The dictionary is rated for children ages 4 – 8 in Amazon.com
reviews, making it an acceptable source of definitions used for young children. An earlier edition of this dictionary was used as a source for building a machine-readable lexical knowledge base represented by Conceptual Graphs (Barriere, 1997). Since the dictionary was previously used for a knowledge base construction, it was assumed to be suitable for the task.

To construct the ontology, each word in the dictionary was represented as an instance of a concept. When two words are synonyms, they are instances of the same concept. The concepts and relationships between them are defined using the formalism of description logic. The formalism chosen for representing knowledge also has to support reasoning because, in addition to knowledge representation, humor detection involves reasoning about knowledge described in a text. Reasoning services vary for different formalisms, as described in the Knowledge Representation (KR) chapter. For this work, a formalism was chosen that allows high expressive power, but does not become undecidable. Since not every natural language expression can be represented with Description Logics, the domain has been restricted to jokes that are not dependent on DL-nonrepresentable knowledge.

Knowledge base in Description Logics consists of two parts: TBox (with sometimes RBox) and ABox. TBox and RBox consist of general knowledge about a particular domain, or intensional knowledge. The knowledge built from the dictionary corresponds to intensional knowledge. ABox contains knowledge about a specific situation, with specific individuals involved. This knowledge is called extensional knowledge. Extensional knowledge is extracted from an input text. An example of TBox knowledge and the corresponding dictionary definitions is shown in Table 3-1. An example of ABox knowledge is shown in Table 3-2.
<table>
<thead>
<tr>
<th>Dictionary definition</th>
<th>DL representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person: A person is a man, woman, boy, or girl</td>
<td>Person $\equiv$ Man $\sqcup$ Woman $\sqcup$ Boy $\sqcup$ Girl</td>
</tr>
<tr>
<td>Woman: A woman is a grown, female person.</td>
<td>Woman $\equiv$ Female $\sqcap$ Person $\sqcap$ Adult</td>
</tr>
<tr>
<td>A girl grows up to be a woman</td>
<td>Girl $\sqcap$ $\exists$temporal.Future $\sqsubseteq$ Woman</td>
</tr>
<tr>
<td>Man: A man is a grown, male person.</td>
<td>Man $\equiv$ Male $\sqcap$ Person $\sqcap$ Adult</td>
</tr>
<tr>
<td>Boys grow up to be men</td>
<td>Boy $\sqcap$ $\exists$temporal.Future $\sqsubseteq$ Man</td>
</tr>
<tr>
<td>Girl: A girl is a female child.</td>
<td>Girl $\equiv$ Female $\sqcap$ Child</td>
</tr>
<tr>
<td>Girls grow up to be women.</td>
<td>Defined above</td>
</tr>
<tr>
<td>Boy: A boy is a male child.</td>
<td>Boy $\equiv$ Male $\sqcap$ Child</td>
</tr>
<tr>
<td>Boys grow up to be men.</td>
<td>Defined above</td>
</tr>
<tr>
<td>Female: Female is a kind of a person or animal. It is the opposite of male.</td>
<td>Female $\sqsubseteq$ Person $\sqcup$ Animal</td>
</tr>
<tr>
<td>Girls and women are female people</td>
<td>Female $\equiv$ $\neg$ Male</td>
</tr>
<tr>
<td>Defined above</td>
<td></td>
</tr>
<tr>
<td>Male: Male is a kind of person or animal. It is the opposite of female.</td>
<td>Male $\sqsubseteq$ Person $\sqcup$ Animal</td>
</tr>
<tr>
<td>Boys and men are male people.</td>
<td>Defined above</td>
</tr>
<tr>
<td>Child: A child is a very young person.</td>
<td>Child $\equiv$ Person $\sqcap$ $\exists$age.VeryYoung</td>
</tr>
<tr>
<td>Boys and girls are children.</td>
<td>Defined above</td>
</tr>
<tr>
<td>Grow up: To grow up means to become a man or a woman. When you grow up, you may be taller than your father.</td>
<td>Defined above</td>
</tr>
<tr>
<td>Cannot be represented</td>
<td></td>
</tr>
</tbody>
</table>

Table 3-1: Dictionary definitions and their DL representation
Mary is a girl.

<table>
<thead>
<tr>
<th>Mary walked to her car.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car(MARY’S_CAR)</td>
</tr>
<tr>
<td>Walked(WALKED)</td>
</tr>
<tr>
<td>agent(MARY, WALKED)</td>
</tr>
<tr>
<td>destination(MARY, MARY’S_CAR)</td>
</tr>
</tbody>
</table>

Table 3-2: Specific knowledge and its DL representation

In Table 3-2, Girl, Car and Walked are concepts, and agent and destination are roles. The roles in the DLs represent relationships between concepts. The concept Girl is defined in Table 3-1. Concepts Car and Walked are defined from dictionary definitions as well. The concepts Car and Girl are children of the concept Object, while the concept Walked is a child of the concept Event. This distinction is important for humor recognition. The Events are treated as scripts for the purposes of this work. Events can occur in the past, present, or future.

The concept Walked is defined differently from the concept Walk, as one has been completed in the past, and the other one may not have. The dictionary contains past tense information only for irregular verbs. Regular verbs were added to the ontology, following the same logical definition as irregular verbs. Adjectives and adverbs were also added to the ontology, as properties of object concepts or event concepts.

It should be noted that actual English words are mapped to concepts as their instances. The instances, of course, are contained in the ABox. Thus, the same TBox can be used for different natural languages. The difference would be introduced by the ABox, which can map word instances from different languages to an already existing conceptual structure.

While the concept hierarchy was built from the dictionary, the relationships design and representation proved to be more difficult. The difficulty was created by natural language ambiguity. For example, the dictionary use of “part-of” could indicate one of following: proper-
part-of-object, part-of-event, component-of, contained-in, member-of. A person can easily select
the proper meaning of “part-of” from the above list using text context, but this task is much more
difficult for a computer, especially if it does not know about the meaning of the relationships in
the list. Such ambiguous relationships needed to be disambiguated for a computer to select the
proper meaning of them. Thus, instead of using the generic “part-of” relationship, its explicit
meaning was used in the created ontology.

The ontology was created using Pellet (Sirin et al., 2005). The use of Pellet simplified
corpus classification and realization of the ontology. The classification feature, supported by
Pellet, automatically computes the classification hierarchy, given defined concepts. Thus, it is
not necessary to specify the exact location of each concept in the hierarchy. The realization
feature computes the direct types of the individuals after classification is performed. Thus, if
MARY is defined as a Girl, and a concept Girl is a subclass of a concept Child and is a subclass
of a concept Female, then MARY is a Child and a Female. The knowledge in the ontology is
represented with the SHI expressivity level of the Description Logics: SHI allows conjunction,
universal value restriction, limited existential quantification, complement, role transitivity, role
hierarchy, and role inverse. SHI does not support number qualification and set concepts, the
features that are supported by SHOIN(D) – a syntactic equivalent of OWL-DL. This restriction
was chosen to improve the apparent speed of the reasoner, and does not significantly affect the
quality of the recognizer. This restriction would have to be removed if the script opposition
would depend on the ontological information contained in the number qualifications or set
concepts, and no workarounds were possible. Unfortunately, the information described in the
dictionary does not provide enough background knowledge to understand most natural language
texts, including jokes. Experiments to detect jokes based on the information described in the
dictionary produced poor results. Upon further analysis, such discovery made sense as the dictionary lacks many of the words used in the tested jokes or the needed meanings of a word used in the joke. As an illustration, consider the following text, taken from a joke book, rated for children ages 7-12:

Text 22: What happened when the ghost asked for a whiskey at his local bar? The bartender said: Sorry, sir, we don’t serve spirits here.

The words “ghost,” “whiskey,” “local,” “bar,” “bartender,” “sir,” “serve,” “spirits” are not in the dictionary. A possible explanation is that this joke is for children 7-12, and the dictionary for an 8-year-old does not have to contain all the words. The same book lists Text 23 as “really funny (if you happen to be around the age of 9)”: 

Text 23: What happens when you drop a piano down a mine? A minor B-flat!

If this joke works for a 9-year-old, and the dictionary is for an 8-year old, it seems that there should not be many missing words. However, “mine” is defined only as a possessive pronoun, which has nothing to do with a mining activity; and, not surprisingly, the words “minor,” “miner,” or “B-flat” are not defined.

It is unreasonable, in the author’s view, to expect humor to be detected without the knowledge of words that trigger humor. Thus, the original ontology was enhanced to include information from a more advanced dictionary. The final version of the ontology contains all knowledge necessary to understand the tested humorous and non-humorous texts.

3.1.2 Finding Semantic Relationships

Finding correct semantic relationships is crucial for humor detection, for both people and computers. People do it subconsciously, but if they fail to find the needed relationships, they
will not find a potentially humorous text to be humorous. The same principle holds for computational detection of humor, provided that the mechanism is semantic in nature. In other words, if the recognizer is based on the meaning of the text, not a combination of the features used in it, it is important to find semantic relationships between words.

Finding correct semantic relationships (or roles) is not an easy task. No computational system is 100% accurate. Punyakanok, Roth, & Yih (2007) report that their system achieved the highest $F_1$ score among 19 participants of CoNLL shared task on semantic role labeling; their website contains a demo for a semantic role labeling task (Cognitive Computation Group). Text 6 (What did the paint give the wall on their first anniversary? A new coat) and Text 8 (What did the wall receive from the paint on their first anniversary? A new coat) were used to test the accuracy of the system. The results of the semantic role labeling are shown in Figure 3-2, as they appear in the semantic role labeling demo.

<table>
<thead>
<tr>
<th>What</th>
<th>thing given (Reference) [R-A1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>did</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>giver[A0]</td>
</tr>
<tr>
<td>paint</td>
<td></td>
</tr>
<tr>
<td>give</td>
<td>V: give</td>
</tr>
<tr>
<td>the</td>
<td>thing given [A1]</td>
</tr>
<tr>
<td>wall</td>
<td></td>
</tr>
<tr>
<td>on</td>
<td>temporal [AM-TMP]</td>
</tr>
<tr>
<td>their</td>
<td></td>
</tr>
<tr>
<td>first</td>
<td></td>
</tr>
<tr>
<td>anniversary</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>What</th>
<th>thing gotten (Reference) [R-A1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>did</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>thing gotten [A1]</td>
</tr>
<tr>
<td>wall</td>
<td>receive V: receive</td>
</tr>
<tr>
<td>on</td>
<td>received from [A2]</td>
</tr>
<tr>
<td>their</td>
<td></td>
</tr>
<tr>
<td>first</td>
<td></td>
</tr>
<tr>
<td>anniversary</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-2: Semantic role labeling output for Text 6 and Text 8 from a demo system (Semantic Role Labeling Demo).
The results show that the system incorrectly identified the semantic role for the phrase *the wall* of Text 6 and incorrectly detected a boundary and therefore incorrectly identified semantic roles of *from paint on their first anniversary*. The system also provides parse trees that were used to select semantic roles, shown in Figure 3-3 and Figure 3-4. The authors argue for the importance of syntactic parsing in semantic role labeling: “full syntactic parsing information is, by far, most relevant in identifying the argument”. The authors’ point of view becomes clear after the parse trees are examined: it is very difficult – if not impossible – to correctly detect semantic roles, when the parse trees that are used as input are incorrect. Unfortunately, no existing system can create syntactic parse trees with perfect accuracy.

(S1 (SBARQ (WHNP (WP What)))
 (SQ (AUX did)
  (NP (DT the)
   (NN paint))
 (VP (VB give)
  (NP (DT the)
   (NN wall))
 (PP (IN on)
  (NP (PRP$ their)
   (JJ first)
   (NN anniversary)))))
 (. ?)))

Figure 3-3: A parse tree used to select semantic roles for Text 6.

(S1 (SBARQ (WHNP (WP What)))
 (SQ (AUX did)
  (NP (DT the)
   (NN wall))
 (VP (VB receive)
  (PP (IN from)
   (NP (NP (DT the)
    (NN paint))
   (PP (IN on)
    (NP (PRP$ their)
     (NN anniversary))))))
 (. ?)))

Figure 3-4: A parse tree used to select semantic roles for Text 8.
A possible explanation for mislabeling the roles could lie in the absence of the preposition to after the verb give. Another text was created in order to test this hypothesis:

Text 24: What did the paint give to the wall on their first anniversary? A new coat.

As shown by Figure 3-5 and Figure 3-6 the addition of preposition to the joke did not affect the correctness of the result.

<table>
<thead>
<tr>
<th>What</th>
<th>thing given (Reference) [R-A1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>did</td>
<td></td>
</tr>
<tr>
<td>the paint</td>
<td>giver[A0]</td>
</tr>
<tr>
<td>give</td>
<td>V: give</td>
</tr>
<tr>
<td>to wall</td>
<td>entity given to [A2]</td>
</tr>
<tr>
<td>on first</td>
<td></td>
</tr>
<tr>
<td>anniversary?</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-5: Semantic role labeling output for Text 24 from the demo system

(S1 (SBARQ (WHNP (WP What))
  (SQ (AUX did)
    (NP (DT the)
      (NN paint))
    (VP (VB give)
      (PP (TO to)
        (NP (NP (DT the)
          (NN wall))
        (PP (IN on)
          (NP (PRP$ their)
            (JJ first)
            (NN anniversary)))))
    (PP (IN on)
      (NP (PRP$ their)
        (JJ first)
        (NN anniversary)))))
  (.) ?)))

Figure 3-6: A parse tree for Text 24 from the demo system
The results of the semantic role selection can be looked at in two ways. The first one can report that out of the three sentences processed, zero sentences received a full accurate semantic role selection. On the other hand, three out of four semantic roles in Text 6 were selected correctly, and two out of four semantic roles were correctly selected in Text 8. If one is interested in the entire meaning of a sentence or a text (and the entire meaning is needed for humor detection), the tool results in 0% accuracy. On the other hand, one can be interested in the accuracy of each individual role assignment, in which case the tool results in 67% accuracy.

For the humor detector semantic relationships are very important. Therefore, a human verifies and corrects semantic relationships proposed by the computer. The computational selection of semantic relationships is based on information about relationships between concepts, provided by the ontology. Whenever a text is read, the words are mapped to instances of ontological concepts. The ontological concepts contain defined relationships between each other. If concept $C_1$ and $C_2$ are triggered by the text, and they have relationships $r_1$ and $r_2$ between them, these relationships are reported as valid semantic relationships.

For example, consider the following sentence: “A man walks on the street.” The following concepts are triggered: $\text{Man, Walk, Street}$, by mapping the word instances $\text{man, walks, street}$ into the corresponding concepts. A concept $\text{Man}$ is a child of a concept $\text{Animal}$. The ontology contains relationship between $\text{Animal}$ and $\text{Walk}$: some $\text{Animal}$ are $\text{agent\_of Walk}$. Since $\text{Man}$ is a child of $\text{Animal}$, the reasoner will infer that $\text{Man}$ is $\text{agent\_of Walk}$. Thus, this relationship is reported as a valid semantic relationship for the text. Similarly, $\text{Walk location Street}$ will be reported.

The function that finds semantic relationships between words is implemented such that if

- Words $w_i, w_j$ occur in a text and
• $w_i$ is an instance of concept $c_i$ and
• $w_j$ is an instance of concept $c_j$, and
• There is an asserted or inferred relationship $r$ between $c_j$ and $c_i$ ($c_j \subseteq \exists r.c_i$ or $c_j \subseteq \forall r.c_i$ or $c_i \subseteq \exists r.c_j$ or $c_i \subseteq \forall r.c_j$)
• Then there is a relationship $r$ between $w_i$ and $w_j$ (either $r(w_i, w_j)$ or $r(w_j, w_i)$).

Since walk is an instance of Walk, man is an instance of Man, and street is an instance of Street, the following entries are valid: location(walk, street); agent_of(man, walk).

All found relationships are presented to a person, who has an option to reject any of them if they are incorrect. The relationships that are accepted are inserted into the ABox. In our example, both relationship are accepted, and are entered into the ABox.

To avoid contradiction, if an instance $w$ of concept $c$ exists in the knowledge base, and a word $w$ occurs in another joke, a new instance $w_1$ of concept $c$ is created. For example, consider a situation where the sentence entered is “A man is asleep”. As in the example above, the following concepts are found: Man, Sleep; and the relationship agent_of. Suppose, it is known that Sleep and Walk are disjoint actions. If the existing instance man is used to represent the man in this scenario, the information from the two statements will result in contradiction. The contradiction makes sense as the sentences either describe two different people, or different times. This works assumes the former, especially, if the same word occurs in two different texts. Under the assumption that two different texts describe two different people, a new instance of concept Man, man1, is created. Then the following is inserted into the ABox: agent_of(man1, sleep), thus avoiding a contradiction.

---

19 Most of the time they are disjoint. The case of sleepwalking will not be considered here.
3.1.3 Humor Recognizer

The humor recognizer component is built to recognize jokes according to the Script-based Semantic Theory of Humor. According to the theory, a text is humorous if it is compatible with two scripts that overlap and oppose. The recognizer treats scripts as children of the Event concept that are defined in the ontology. These children are referred to as events in this work. Each event can contain sub-components that are indicated by the DL role “part-of-event”. The events also have a DL role “effect” which indicates the end result of the event, or its goal. These end results or goals are also concepts, and are referred to as event effects or effects. Some events have multiple effects.

The words from the text are mapped as instances to the concepts. Each word that corresponds to an instance of an event triggers this event. For example, the sentence “A man walks on the street” contains one script: Walk. The word walk triggers the script (or event) Walk.

Two scripts (events) are considered overlapping if they have at least one concept in common that is part of both events. In the sentence “A man walks into a bar”, there are at least two scripts, roughly\(^{20}\) corresponding to Moving on foot and Hitting (or being hit by) something. These scripts are triggered by the word walk (into) which can be mapped as an instance of both of the above events. The concept Man, which has an instance man, overlaps both events. It plays the role of a semantic agent, which has the corresponding DL role “agent” in the ontology, in the Moving on foot event; and, arguably, a beneficiary role in the Hit event. The word bar, on the other hand, does not create a semantic overlap because it corresponds to two different

\(^{20}\)The word *roughly* is intended in its common use here, not as a technical term in Rough Sets.
concepts for the two scripts (roughly, concepts Restaurant and Pole). The overlap here is orthographic and phonological. Formally, two scripts (events) overlap if the intersection of their components is not empty. A component is defined as any concept accessed in and by the event, whether it is by default or specified in the text.

Script opposition is usually defined as local antonymy (see Section 2.4.1 for more discussion). To use this definition, the grain size of the scripts has to be relatively coarse. Opposition is used as disjointness to find an antonymous event instead of climbing the parent hierarchy. The use of disjoint events obviates the problem of opposition being treated as negation. For example, the meanings of words like “old” and “young” are considered disjoint. Similarly, the meaning of “asleep” and “awake” are considered disjoint. A less obvious example is perhaps that “physical object” and “abstract object” are disjoint concepts. The disjointness comes from considering some salient properties of the concepts. Intuitively, two concepts A and B are disjoint (on the same semantic plain) if fuzzification (Zadeh, 1965) of all concepts of this semantic plain produces no fuzzy overlap between A and B. Figure 3-7 shows four sets (such as baby, child, adult, senior) defined though one property (such as age). The concepts A and C, A and D, B and D are disjoint, while A and B, B and C, C and D are not.

![Figure 3-7: Disjointness of concepts through their properties](image)

The concepts, of course, can be defined in terms of several different properties.
For the purposes of this research, script opposition was defined not only as disjointness of events, but as disjointness of the effects or goals of the events. In other words, while *Walking into a bar* is not necessarily disjoint with *Hitting a pole* – they may be non-overlapping, but not necessarily disjoint on the same semantic plane – *walking into a bar for the purposes of having a good time* (result: enjoyment) and *being hit by something* (result: pain) do have a common semantic plane. It is unclear if script opposition can be defined as disjointness of effects for a more general case, but it worked well as a heuristic function in this investigation.

Because the humor recognizer determines its results based on the information contained in the knowledge base, it is able to not only generate its answer as to whether a text is humorous, but also provide an explanation for the result. It is able to show what concepts overlap semantically, what words trigger orthographic but not semantic overlap, and what events have disjoint effects.

To summarize, the humor recognizer returns that a text is humorous if it finds two events that have some concepts in common, and the events’ effects are disjoint.

### 3.2 Phonological Component

Some humorous texts are produced by words that sound alike or similar (with different spelling). These texts have the same mechanism of humor recognition as texts with ambiguous words, once the similar-sounding words are identified and processed. Consider the following text:


The semantic component by itself will not be able to find script overlap / opposition because without the phonological component there is no connection from a fish, sturgeon, to a medical doctor, surgeon. Once this connection is found, the resulting events are surgery (or operation)
and existence (life) of fish. The first event has an effect of healing (in this case a purposeful action), while the second one has a passive existence, which is not a purposeful action. Thus, the effects are disjoint. Moreover, a fish is not capable of performing a purposeful action, such as an operation, which, again, creates script opposition from the knowledge that we have about fish, and what it actually does according to the this text. The overlap here is that the agent of the Surgery script or a beneficiary of the Existence script is the fish.

A method has to be found to express the mapping of sturgeon to surgeon in order to make the script overlap / opposition, described in the paragraph above, work. In order to find an acceptable mapping in the text, several questions have to be addressed:

• What word should be replaced with a phonologically similar word?

• What phonologically similar word should be chosen from the list of all possible words?

The first question is addressed in the orthographic component section; in this section the concentration is on the second question.

The phonological representation of words was taken from the CMU Pronouncing Dictionary. The dictionary contains over 125,000 words and their transcriptions for North American English.

In order to find similarly sounding words, a measure for similarly sounding phonemes has to be established. In this dissertation the similarity of phonemes is taken from Hempelmann (2003) who addresses the cost of changing a phoneme in the source-target pair of a pun. In Text 25, “sturgeon” is the source, and “surgeon” is the target. The cost coefficients c are shown in Appendix A. The cost takes into account environmental factors (such as syllable position, neighboring segments), IDENT constraint violations (such as level and type of a feature), and suprasegmental factors (such as change in stress position, or deletion/insertion of a syllable):
\[ c = \frac{\sum_{i=1}^{n} v \times e + \alpha + \sigma}{xs} \]

where \( v \) is the IDENT constraint violations; \( \sigma \) is the number of deletion/insertion of a syllable; \( \alpha \) is the number of changes in the stress position; \( e \) is the environmental factor, \( s \) is the number of segments; and a factor \( x \). Identical phoneme pairs have the cost of 0, while those that are unrecoverable have a cost of 1.

There are several ways to find words that are phonologically similar. The input word is referred to as the source, and the word to be found as the target. The two simplest ways to find similarity (or cost) between the source and the target are:

- **Replacement of each phoneme of the source, until a desired similarly sounding target is found.** Assuming that the number of phonemes in a source is \( v+c \), where \( v \) is the number of vowels, \( c \) is the number of consonants, if no phonemes are deleted or added, in the worst case scenario, there are \( 24^c \times 17^v \) generated strings for a target selection\(^{21}\). Each resulting phonological string should be compared to the existing words in the CMU dictionary -- only real word targets are used. Assuming that the query engine uses an efficient search algorithm, the resulting formula is \( \log_2 n \times 24^c \times 17^v \), where \( n \) is the number of words in the CMU dictionary, \( c \) is the number of consonants and \( v \) is the number of vowels in the original phonological string.

- **Comparison of source with each word in the CMU dictionary.** As in the previous method, for this investigation, the source and the target are considered to be of the same length. Assuming that the length of the source is proportional to the number of dissimilar

\(^{21}\) There are 25 consonants and 18 vowels, according the cost table (Hempelmann, 2003).
phonemes between the source and the recoverable target; and limiting this number to 
\( \text{MIN}(3, 0.5 \times \text{length of the source}) \), in the worst case scenario, \((n-1) \times (\text{min}(3, 0.5(v+c)))\) calculations are made, where \(n\) is the number of words in the CMU dictionary, \(c\) is the number of consonants and \(v\) is the number of vowels in the original string.

In this investigation, the second method was used. The decision was based on the worst-case number of calculations. The results of the comparisons of the two methods are not affected if the source and the target can have different lengths. However, the algorithm used for finding the target allows for it.

3.2.1 Similar Sounding Word Detector

When a word, referred to as source, is entered into the Similar Sounding Word Detector, a query to the CMU database is generated that produces a phonological representation of the source. The second query to the database results in a list of phonological representations of words, such that the number of different phonemes between the source and each entry in the list is no more than 3 and no more than half of the length of the phonological representation of the source.

While the complexity calculations assumed that the number of phonemes between the source and the target is the same, the Similar Sounding Word Detector allows for the phonemes to be dropped or added. The deleted or added phonemes contribute to the different phoneme count. Because of the added and dropped phonemes, a simple comparison of \(i^{th}\) element between the two arrays will not produce the desired result. For example, comparing the words “cat” and “scat” should result in one phoneme difference. But, if we compare the first element of “cat” to the first element of “scat”, the second element of “cat” to the second element of “scat”, etc., the result will indicate 4 phonemes that differ. To overcome this limitation, a Levenshtein distance
The (Levenshtein, 1965) algorithm was selected. The algorithm calculates the number of character additions, replacements or deletions needed to transform one string into another. The algorithm was modified to take into account cost, ranging from 0 to 1, of phoneme modification, as indicated by the Hempelmann table, instead of using the constant cost of 1 for all changes.

For any two words, the cost of reaching the target from the source was calculated. A list of possible targets was constructed for every possible source, sorted in ascending order by the cost. The ordered list of words was passed to the orthographic component for further processing.

### 3.3 Orthographic Component

The orthographic component serves as the input into the humor detector: it reads the text, queries words’ familiarity and frequency, and maps words into ontological concepts.

The words are mapped into concepts as their ontological instances. If a word has more than one meaning, it is mapped into a disjunction of concepts representing each possible meaning of the word. For example, a word *child* can mean an “offspring” (concept Offspring), regardless of the child’s age, or a young – lets say, under 13 years old -- human (concept Child). The word *child* will then become an instance of a concept Offspring ∪ Child. This indicates the word can have either meaning.

The metrics for familiarity and frequency of words play an important role in detection of jokes that are based on phonological similarity. These metrics are used for a reduction of space of possible targets or sources.

#### 3.3.1 Familiarity and Frequency of Words

There are seven words in the sturgeon joke. Potentially, each of them could be replaced with a phonologically similar word (with the exception of the word “a”), as shown in Figure 3-8. The
combinations of words are not considered for replacement due to the experimental design restrictions.

Figure 3-8: Number of word replacements

To minimize the number of unnecessary word replacements and comparison, several heuristic functions are used. The first heuristic function is based on the fact that the punchline of a joke is usually at the end. Therefore, it is reasonable to assume that the word to be replaced is found closer to the end of the text. Using this intuitive approach, the words in the text are considered in the “last in, first out” order. This means that the word “sturgeon” is the first replacement consideration.

The second heuristic function is based on observation that the words that people recover are unlikely to be unfamiliar or infrequently used. In Taylor and Mazlack (2007), we tested whether there was a relationship between familiarity or frequency measures of the source and target in puns. The results showed that in most puns the source had lower familiarity value than that of the target, and lower frequency value than the target. The results are used to reduce the number of words in the list that are received from the phonological component prior to testing these
words in the semantic component. Namely, the target words that have familiarity and frequency lower than the source are removed.

The third heuristic function is used to restrict the source selection. This function looks at the words in the text before the phonological component is called. The hypothesis is that the source has a familiarity and frequency values that are lower than most words in the joke. The hypothesis was tested on puns in Taylor and Mazlack (2007). Thus, the number of potential sources of the pun can be reduced, based on the familiarity and frequency values. Each source in the reduced list is passed to the phonological component in the order selected by the first heuristic function until the joke is detected, or the list is empty.
4 EXPERIMENTS

The question to be answered by this dissertation is whether computational detection of wordplay humor is possible with a meaning-based approach. Moreover, the interest is not in the classification result by itself (joke vs. non-joke) but also in the explanation, provided by the machine, of why the classification has been made. In other words, the black box, shown in Figure 4-1, is the center of attention in this dissertation.

Figure 4-1: Abstraction of humor detector

Inside the black box is the decision mechanism that produces the output. The detection decision mechanism in this dissertation relies on existing semantic humor theories. While these theories do not contain precise algorithms for humor detection themselves, they provide enough information for computational algorithm construction.

The theories that served as foundation for this work, and were modeled, are semantic in nature. Thus, for the modeled theories to be applicable, a computer must detect the meaning of an input text, including the meaning of words and utterances inside the text. If a computer has no knowledge of a word that is used in a joke, or of its meaning, it is unreasonable to assume that a joke can be detected or understood. Once the meanings of words are available, and possible interpretations are made, humor theories are used to conclude if the text is humorous. Therefore, the experiments take into account:
(A) Whether the ontology contains semantic knowledge about the words used in a joke.
(B) Whether the needed meaning of the word is defined in the ontology.
(C) Whether the ontology contains the needed relationships between concepts, represented by words in the joke.
(D) Whether the joke can be recognized when (A), (B) and (C) are satisfied.

These questions are answered in order to better explain the results produced by the “black box”.

The aim of this dissertation is to recognize wordplay-based humor in short texts written for young children. Specifically, the goal is to determine if humor detection is possible in wordplay jokes, where the utterance order in not restricted by known precise templates. The focus is on the meaning of text, not its structure. The central hypothesis is that humor recognition of natural language texts is possible when knowledge needed to understand the texts is available in a machine-understandable form.

The central hypothesis is tested by experiments designed to answer the following five questions:

- Is it possible to recognize jokes that are based on word ambiguity?
- Is it possible to recognize jokes that are based on phonological similarity of words?
- Can jokes be recognized by comparing them with already known jokes?
- Can jokes be recognized when an ontology does not have all of the required background knowledge to process the meaning of text?
- Are some jokes easier to recognize than others?
4.1 Samplle Size

The sample size is chosen so that an analysis of variance can be performed on the results of the experiments. The experiments are described later in the section. It is assumed that the jokes and non-jokes are randomly selected within the predefined criteria. In other words, if the criterion is children’s jokes involving animals, the sample jokes are randomly selected within all children’s animal jokes.

Avoiding Type-I and Type-II errors, and the effect size\(^{22}\) are considered in the selection of a sample size.

Both joke and non-joke groups are of an equal size. Since the recognizer is considered successful if recognition of correct type is better than random, it can be assumed that a small deviation is still acceptable. Therefore, the effect size can be small.

If recognition of the correct type is random, half of the jokes and half of the non-jokes can be expected to be recognized correctly; then, \(|\mu_{joke}-\mu_{non-joke}|=0.5\) and \(|\mu_{joke}-\mu_{non-joke}|=0\). For a successful recognizer, \(|\mu_{joke}-\mu_{non-joke}|>0\); for the sample size calculation purposes, \(|\mu_{joke}-\mu_{non-joke}|>0.1\).

Type-I error means that the null hypothesis is rejected, even though it is true. Common values for \(\alpha\), or \(P(\text{Type-I error})\), are 0.01, 0.05, 0.1. Since implications of Type-I error for this research are not very serious, the \(\alpha\) value is chosen to be 0.05.

Type-II error means that the null hypothesis is not rejected even though it is false. As with Type-I error, \(P(\text{Type-II error})\), or \(\beta\), should be reasonably minimized. Typically, the values for

\(^{22}\) The is a measure of the strength of the relationship between two variables (jokes and non-jokes).
P(Type-II error) range from 0.01 to 0.2. Again, since the implications of Type-II errors for this research are not very serious, the $\beta$ value is chosen to be 0.2.

The sample size also depends on standard deviation. The standard deviation, $\sigma$, cannot be more than 0.5. Using the values $\beta=0.2$, $\alpha=0.05$ and effect=0.1, the required sample size is 197 when $\sigma=0.5$; the required sample size is 126, when $\sigma=0.4$; the required sample size is 71 when $\sigma=0.3$.

The sample size of 200 (100 jokes, 100 non-jokes) is chosen to accommodate the largest possible standard deviation for this experiment.

4.2 **Joke Selection**

A total of 100 jokes and 100 non-jokes were tested. The jokes were selected by three volunteers not familiar with the details of this dissertation. They are native speakers of English, with a high school diploma, and in their opinion (no formal tests were performed) have a good humor appreciation. These people were asked to find jokes similar in form and topic to example jokes culled from sources suggested by the TeachersVision website. The website provides lesson plans for preschool and elementary school teachers. Specifically, the following explanation is given for the suggested sources: (http://www.teachervision.fen.com/tv/printables/TCR/1420633929_060-062_d.pdf):

Elementary-school children are at an age where they are beginning to have a better command of words and language. Many jokes for children are funny because the punchline is a play on words. Students this age are just beginning to appreciate and “get” the joke because of their increasing understanding of the English language.

The jokes taken from the suggested sources can be divided into three categories:
(A) Jokes that are based on phonological similarity between two existing words
(different spelling, similar pronunciation)

(B) Jokes that are based on word ambiguity (same spelling)

(C) Jokes that are based on other factors

Only jokes that are based on (A) and (B) were considered for this dissertation. The volunteers that selected jokes were given an additional restriction for joke selection. Namely, jokes satisfying any of the following three criteria should not be considered: (a) containing made-up words, (b) containing phonological similarity between phrases, rather than individual words, and (c) containing non-literal language on which the joke understanding depends. While (a) and (b) were easy to follow, whether a joke violates (c) is a matter of opinion. For this dissertation, an opinion of the volunteer was considered primary. Thus, there may be jokes in the joke set that seem to violate the above restrictions. For example, Text 27 was selected by one of the volunteers. The joke violates the (b) criterion: an arrow --> a narrow. However, it was included in the joke set.

Text 26: What did Robin say when he nearly got hit at the archery contest? "That was an arrow escape!"

The suggested sources divide jokes into themes. The themes were: animals and pets, doctor and nurse, scary and monster, school humor. Thus, categories (A) and (B) were further divided into these themes.

100 jokes were selected such that: 50 jokes were based on category (A) and 50 jokes were based on (B). All 100 jokes should fit into five themes such that each theme contained 10 jokes from category (A) and 10 jokes from category (B). The original themes were taken from sources suggested by the TeacherVision website. An original theme could remain the same, be split into
two finer grain themes, or two themes could be combined to a single larger grain theme. All the themes were created by the volunteers that selected the jokes. They were given instructions to make up five themes that would encompass as many of the 100 jokes as possible. The jokes that did not fit were replaced with the new ones.

The resulting five joke themes are:

- fairytale jokes
- monster jokes
- mammal jokes
- non-mammal animal jokes (insects, fish, birds)
- people jokes (doctor and school jokes)

The resulting joke grid is shown in Table 4-1.

<table>
<thead>
<tr>
<th></th>
<th>Fairytale</th>
<th>Monster</th>
<th>Mammal</th>
<th>Non-mammal animal</th>
<th>People</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Ambiguity</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>Phonological Similarity</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>Total</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 4-1: Categorization of jokes to be tested

4.3 **NON-JOKE TEXTS**

Additionally, 100 non-joke texts were artificially created. Each text of the non-joke category is based on one of the tested jokes. The only difference is that the non-jokes do not contain words that trigger script overlap or script opposition. As an example, consider Text (a joke based on word ambiguity) and Text (a non-joke):

Text 27: What do you call a witch that climbs up walls? Ivy!
Text 28: What do you call a witch that climbs up walls? Mary!

Text is a joke because the word “Ivy” can mean the name of a witch, as well as a plant that climbs up walls. Thus, two scripts are activated: a climbing person and a growing plant. Assuming that the person gets to the top of the wall, or at least to the desired point, the effect of the climb is the successful finish of the purposeful action. The effect of a growing plant could hardly be considered a purposeful action, at least not by the plant. Thus, the effects of two scripts are in opposition. The script overlap is in naming the actors corresponding to both events, as well as the spelling of the actors (the plant and the witch) and the events. If these scripts are found, Text is considered a joke.

Text, on the other hand, cannot activate two scripts. “Mary” is a name of a person, but not a name of a plant, or any other commonly known entity that can climb the walls. This means that the only available script is a climbing person. Thus, script overlap and opposition will not be found, and the text should not be considered a joke.

The non-jokes can be organized in a similar fashion as jokes, as shown in Table 4-2.

<table>
<thead>
<tr>
<th></th>
<th>Fairytale</th>
<th>Monster</th>
<th>Mammal</th>
<th>Non-mammal animal</th>
<th>People</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Removed Ambiguity</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>Removed Phon. Sim.</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>Total</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 4-2: Non-jokes, created from jokes by removing word ambiguity or phonological similarity.
4.4 **Joke Recognition: What and Why**

The subgoal of this dissertation is to provide an explanation of why a joke is detected or not detected. Thus, simple joke or non-joke classification is not enough. The jokes are considered to be successfully recognized only when script opposition is correctly identified.

As stated in Section 2.4.3, some texts are commonly referred to as puns when they contain wordplay without a found script overlap and opposition. These texts are expected to be referred to as “wordplay-based non-jokes” by the detector, not “jokes”. In other words, if there is a text, which contains either word ambiguity or phonological similarity, and the ambiguous meanings or meanings of the similar sounding words are defined in the ontology as disjoint, the text is considered a “wordplay-based non-joke”.

The joke books and websites do not distinguish between “jokes” and “wordplay-based non-jokes”. Therefore, every text in the 100 joke set can be either a real “joke”, or a “wordplay-based non-joke”. The identification is considered successful in both cases. The distinction is made only for the analysis purposes.

Because the detection mechanism is important, and detection depends on the ontological knowledge, the following five questions are addressed:

(A) Does the ontology contain all the words in a joke?

(B) Are all the needed meanings of each word defined in the ontology?

(C) Are all the needed meanings of the salient words defined in the ontology (words that trigger script overlap and script opposition)?

(D) Does the ontology contain the needed relationships between concepts, represented by words in the joke?

(E) Can the joke be recognized when (A), (B), (C) and (D) are satisfied?
The (A), (B), (C), (D) address how much knowledge is necessary for a computer to detect a joke, while (E) addresses the general problem of whether detection of the joke is possible. In other words, (A), (B), (C) and (D) can be used to analyze whether all words in the joke and their relationships are crucial to joke detection. Intuitively, a successful humor detector should have a high accuracy when (E)\(^23\) is satisfied; and, it should be able to successfully detect jokes when (C) and (D) are satisfied. It could be argued that when (C) is not satisfied, it is unreasonable to expect a correct classification of texts, and an explanation of why the text is classified as joke or non-joke.

### 4.5 EXPERIMENTS

Several experiments were conducted to answer the questions that test the central hypothesis of this research. The experiments’ design, their sample size, and theoretical foundations of result interpretations are discussed in this section.

#### 4.5.1 When is the experiment successful?

A Bernoulli trial process was used to interpret the results of the experiments. A Bernoulli trial process consists of a series of experiments, with the outcome of success or failure. The experiments are independent of each other, and have the same success probability, \(p\). The probability of failure is \((1-p)\).

The recognizer is considered successful if its performance is better than chance. A chance performance is defined as probability of success of a fair coin (lands heads), which, by

---

\(^{23}\) We are assuming that the theory and implementation are correct.
definition, is 0.5. The expected value of chance success performance of the humor detector on each joke is 0.5.

The standard deviation can then be calculated as \( \sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2} \). The detector is considered successful, and each question is successfully answered if the result is at least 2\( \sigma \) away from the mean. On a sample of texts when \( n=100 \), \( \sigma^2=100*(0.5-0.5^2)=100*0.25=25 \), \( \sigma=5 \), thus the recognizer is successful if at least 60 jokes are recognized (its success rate is above 60%). On a sample of texts when \( n=50 \), \( \sigma^2=50*(0.5-0.5^2)=50*0.25=12.5 \), \( \sigma=3.5 \), thus the recognizer is successful if \( 25+3.5*2=32 \) jokes are recognized (its success rate is above 64%).

4.5.2 Is it possible to recognize jokes that are based on word ambiguity?

The first question to be answered is: Is it possible to recognize jokes that are based on word ambiguity? To answer the question, 50 jokes that are based on word ambiguity and 50 non-jokes, generated from the jokes, are tested. For each of the jokes and non-jokes the answers to questions (A) – (E) are recorded. The detector is considered successful if one of the following is true:

- For all texts that satisfy (E), random classification is worse with statistical significance than a classification by the humor detector; and the explanation of the classification is acceptable.
- For all texts that satisfy (C) and (D), random classification is worse with statistical significance than a classification by the humor detector; and the explanation of the classification is acceptable.

The question can be positively answered if the recognizer is considered successful.
4.5.3 Is it possible to recognize jokes that are based on phonological similarity of words?

The second question to be answered is: Is it possible to recognize jokes that are based on phonological similarity of words? To answer the question, 50 jokes that are based on phonological word similarity and 50 non-jokes, generated from the jokes are tested. Similar criteria are used to answer this question, as the previous question with the addition of the following:

(F) was the needed wordplay found?

The detector is considered successful if one of the following it true:

• For all texts that satisfy (E) and (F), random classification is statistically significantly worse than a classification by the humor detector; and the explanation of the classification is acceptable.

• For all texts that satisfy (C), (D) and (F), random classification is statistically significantly worse a classification by the humor detector; and the explanation of the classification is acceptable.

4.5.4 Can jokes be recognized by comparing them with already known jokes?

The third question to be answered is: Can jokes be recognized by comparing them with already known jokes? To answer this question, 50 previously detected jokes and 50 new jokes are tagged with the knowledge resources from the General Theory of Verbal Humor (GTVH). The theory is used to compare previously used jokes and the new jokes. All 50 new jokes must be similar to the 50 previously detected jokes, according to the GTVH criteria. A volunteer, familiar with the GTVH, tagged the jokes, according to the knowledge resources. For this
experiment, the jokes are considered similar, if they contain the same SO and the same LM, as tagged by a human. The GTVH tags are not available to a computer. The 50 new jokes must satisfy at least (A) and (B). The detector is considered successful if the classification and explanation of the results is statistically better than chance.

### 4.5.5 Can jokes be recognized when an ontology does not have all of the required background knowledge to process the meaning of text?

The fourth question to be answered is: Can jokes be recognized when an ontology does not have all of the required background knowledge to process the meaning of text? This means that (A), (B) or (C) is satisfied, but (D) is not. A positive answer to the question is given if the detection of jokes when (D) is not satisfied is statistically better than random.

This question has been added after the dissertation proposal was developed. The motivation for this question is to understand whether all semantic relationships and background information are equally important. In other words, can a joke be detected when some information needed to fully understand the joke is not there? Consider the following as an example:

**Text 30:** Ali Baba didn't know it, but there were four women locked in the cave with all that jewelry. What were their names? Ruby, Jade, Coral and Pearl!

To fully understand the joke, one needs to know what is “ruby”, what is “jade”, what is “coral”, and what is “pearl”. In addition to that, one must understand that Ruby, Jade, Coral, and Pearl are all women’s names, and therefore, all four of them are people. This means that any of these words can (and do) mean an animate and inanimate object at the same time, therefore, the text is a wordplay. The interesting part here is that it is sufficient for one of the four words to be recognized as animate and inanimate, the other three do not have to be detected (at least as far as
this detector is concerned) for it to be a joke. The question is, how often does it happen? What proportion of the jokes is recognized without containing all relevant information? Is this proportion higher than random?

4.5.6 Are some jokes easier to recognize than others?

The fifth question to be answered is: Are some jokes easier to recognize than others? The five themes of jokes are compared with respect to the detection success. To answer this question, a null hypothesis is introduced:

\[ H_0: \text{there is no difference in identification of jokes and non-jokes between themes} \]

A 2(joke vs. non-joke) x 5(number of themes) analysis of variance is be performed. The critical value of \( F(4,190)=2.37 \), when \( \alpha=0.05 \). The null hypothesis is be rejected if \( F>2.37 \).

4.5.7 Other Proposal Questions

There were several questions in the proposal that this dissertation does not address. The questions are:

- Is there a relationship between joke recognition and semantic distance between concepts within a joke? Is there a semantic distance threshold for joke recognition?
- Does phonological distance between concepts play a role in joke recognition? Is there a phonological threshold for joke recognition?
- Is there a relationship between the grain of semantic tagging and joke/non-joke identification?
- Is statistical knowledge a useful tool for joke recognition?
- Finally, is ontological recognition of jokes more successful than statistical joke
It was discovered during the course of research that while these questions can be answered they don’t shed light on the problem at hand. For example, the semantic distance between concepts is useful only when the network structure that it relies on is complete and does not require any change. Unfortunately, it is not the case with ontologies representing general knowledge: concepts are regularly added and removed. Since semantic distance is measured in terms of links (edges in a graph) between concepts, such additions or removals result in modification of the semantic distance. Yet, the difference between a cat and a dog should be the same, no matter how many links they have between them.

Similarly, phonological distance and threshold are not considered. This would be a useful experiment if the phonological distance was the only factor contributing to source recoverability. However, there are others, as shown by the experiments with semantic priming, described earlier.

The grain of the semantic tagging question was not answered because no tagging *per se* was involved in the humor detection. The meaning of the words was detected without being tagged. It could be interpreted that the detector selected the best available grain, and therefore, it could be compared against something. However, the grain would be measured in terms of the links, which is unlikely to provide useful information.

Finally, the question of statistical information is only meaningful when there is no explicit knowledge about something available. For this research, detecting the meaning of words, using statistical information, when it is possible to do it with explicitly defined concepts in the knowledge base, does not make sense.
5 RESULTS

One hundred jokes and one hundred non-jokes were used to test the central hypothesis: humor recognition of natural language texts is possible when knowledge needed for humans to understand the texts is available in a machine-understandable form.

5.1 IS IT POSSIBLE TO RECOGNIZE JOKES, BASED ON WORD AMBIGUITY?

One of the questions to be answered to test the central hypothesis is: is it possible to recognize jokes that are based on word ambiguity? Fifty jokes and fifty non-jokes were used to answer this question. Table 5-1 shows the number of texts (jokes and non-jokes) correctly identified, when all words in the text were known to the detector.

<table>
<thead>
<tr>
<th></th>
<th>Word Knowledge</th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes</td>
<td>All words known</td>
<td>22%</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>Not all words known</td>
<td>6%</td>
<td>68%</td>
</tr>
<tr>
<td>Non-jokes</td>
<td>All words known</td>
<td>0%</td>
<td>26%</td>
</tr>
<tr>
<td></td>
<td>Not all words known</td>
<td>0%</td>
<td>74%</td>
</tr>
</tbody>
</table>

Table 5-1: Known words from *The American Heritage First Dictionary* (2006)

Table 5-2 shows the number of texts that contain word meanings that were actually used in texts, and texts identification as jokes or non-jokes.

<table>
<thead>
<tr>
<th></th>
<th>Meaning Knowledge</th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes</td>
<td>All needed meanings are known</td>
<td>16%</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>Not all needed meanings are known</td>
<td>12%</td>
<td>68%</td>
</tr>
<tr>
<td>Non-jokes</td>
<td>All needed meanings are known</td>
<td>0%</td>
<td>26%</td>
</tr>
<tr>
<td></td>
<td>Not all needed meanings are known</td>
<td>0%</td>
<td>74%</td>
</tr>
</tbody>
</table>

Table 5-2: Known meanings from *The American Heritage First Dictionary* (2006)
Table 5-3 shows the number of texts that were identified as jokes or non-jokes and their relationship to meanings of salient words or concepts, known to the computer.

<table>
<thead>
<tr>
<th></th>
<th>Salient Knowledge</th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Jokes</strong></td>
<td>All salient meanings are known</td>
<td>22%</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>Not all salient meanings are known</td>
<td>6%</td>
<td>68%</td>
</tr>
<tr>
<td><strong>Non-jokes</strong></td>
<td>All salient meanings are known</td>
<td>0%</td>
<td>26%</td>
</tr>
<tr>
<td></td>
<td>Not all salient meanings are known</td>
<td>0%</td>
<td>74%</td>
</tr>
</tbody>
</table>

Table 5-3: Salient meanings needed for the joke detection from *The American Heritage First Dictionary* (2006)

Table 5-4 shows the number of texts that were identified as jokes or non-jokes corresponding with the number of texts with conceptual relationships known by the ontology.

<table>
<thead>
<tr>
<th></th>
<th>Relationship Knowledge</th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Jokes</strong></td>
<td>All needed relationships are known</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Not all needed relationships are known</td>
<td>28%</td>
<td>72%</td>
</tr>
<tr>
<td><strong>Non-jokes</strong></td>
<td>All needed relationships are known</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Not all needed relationships are known</td>
<td>0%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 5-4: Relationships between concepts needed for joke detection from *The American Heritage First Dictionary* (2006)

The results in the tables do not look promising. There is clearly not enough background information in the dictionary that can be used to produce semantic relationships. The dictionary that was used to produce these results contains only about 1800 entries. The question arises: can these results be improved if a dictionary with more entries is used? To answer this question, *The
Dorling Kindersley Children’s Illustrated Dictionary (McIlwain, 1994) was used. This dictionary contains around 5000 words and is rated for children aged 5 to 8. The ontology was updated with the information from this dictionary. This updated ontology is referred to as ontology2. The same experiments were performed. The results are shown in Table 5-5, Table 5-6, Table 5-7 and Table 5-8.

<table>
<thead>
<tr>
<th></th>
<th>Word Knowledge</th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes</td>
<td>All words known</td>
<td>38%</td>
<td>26%</td>
</tr>
<tr>
<td></td>
<td>Not all words known</td>
<td>8%</td>
<td>28%</td>
</tr>
<tr>
<td>Non-jokes</td>
<td>All words known</td>
<td>8%</td>
<td>28%</td>
</tr>
<tr>
<td></td>
<td>Not all words known</td>
<td>0%</td>
<td>64%</td>
</tr>
</tbody>
</table>

Table 5-5: Known words from *The Dorling Kindersley Children’s Illustrated Dictionary* (McIlwain, 1994)

<table>
<thead>
<tr>
<th></th>
<th>Meaning Knowledge</th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes</td>
<td>All needed meanings are known</td>
<td>26%</td>
<td>26%</td>
</tr>
<tr>
<td></td>
<td>Not all needed meanings are known</td>
<td>20%</td>
<td>28%</td>
</tr>
<tr>
<td>Non-jokes</td>
<td>All needed meanings are known</td>
<td>8%</td>
<td>38%</td>
</tr>
<tr>
<td></td>
<td>Not all needed meanings are known</td>
<td>0%</td>
<td>54%</td>
</tr>
</tbody>
</table>

Table 5-6: Known meanings that are needed for joke detection from *The Dorling Kindersley Children’s Illustrated Dictionary* (McIlwain, 1994)
<table>
<thead>
<tr>
<th></th>
<th>Salient Knowledge</th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes</td>
<td>All salient meanings are known</td>
<td>26%</td>
<td>26%</td>
</tr>
<tr>
<td></td>
<td>Not all salient meanings are known</td>
<td>20%</td>
<td>28%</td>
</tr>
<tr>
<td>Non-jokes</td>
<td>All salient meanings are known</td>
<td>8%</td>
<td>38%</td>
</tr>
<tr>
<td></td>
<td>Not all salient meanings are known</td>
<td>0%</td>
<td>54%</td>
</tr>
</tbody>
</table>

Table 5-7: Salient meanings needed for the joke detection from *The Dorling Kindersley Children’s Illustrated Dictionary* (McIlwain, 1994)

<table>
<thead>
<tr>
<th></th>
<th>Relationship Knowledge</th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes</td>
<td>All needed relationships are known</td>
<td>4%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Not all needed relationships are known</td>
<td>42%</td>
<td>24%</td>
</tr>
<tr>
<td>Non-jokes</td>
<td>All needed relationships are known</td>
<td>0%</td>
<td>26%</td>
</tr>
<tr>
<td></td>
<td>Not all needed relationships are known</td>
<td>8%</td>
<td>66%</td>
</tr>
</tbody>
</table>

Table 5-8: Relationships between concepts needed for joke detection from *The Dorling Kindersley Children’s Illustrated Dictionary* (McIlwain, 1994)

The comparison of the corresponding tables shows that the results have improved. This suggests that the more relevant knowledge the ontology has, the more accurate the humor detector is. Table 5-9 shows the results of the humor detector when the ontology contains all the necessary knowledge needed to understand texts (jokes or non-jokes).

<table>
<thead>
<tr>
<th></th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes</td>
<td>64%</td>
<td>36%</td>
</tr>
<tr>
<td>Non-jokes</td>
<td>8%</td>
<td>92%</td>
</tr>
</tbody>
</table>

Table 5-9: Text detection when all required information is known

Table 5-9 can be used to answer the question whether the computational detection of jokes, that are based on word ambiguity, is possible. The table shows that the recognition of jokes as
such was at least two standard deviations away from chance results. The recognition of non-jokes as such was also at least 2 standard deviations away from the results obtained by chance (as described in the Experimental Design section). The combined identification of jokes and non-jokes was also successful \(((64\%+92\%)/2=78\%)\).

### 5.2 IS IT POSSIBLE TO RECOGNIZE JOKES, BASED ON PHONOLOGICAL WORD SIMILARITY?

To answer this question, fifty jokes and fifty non-jokes were used. The difference between the ambiguity-based class of jokes and this class of jokes is the ability of the detector to find the word that sounds similar to some word in the joke. This problem is three-fold. First, the word that should be replaced must be identified. Second, a word, sounding similar to the to-be-replaced word, must be found. Third, script overlap and script opposition (SO) must be identified. The detector was very successful at the first task (96% success rate). It was to a smaller degree successful with the second task (76% success rate). Table 5-10 shows the results after completing all tasks.

<table>
<thead>
<tr>
<th>Similar sounding words trigger SO</th>
<th># of correctly detected words to be replaced</th>
<th># of unexpectedly detected words to be replaced</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>54%</td>
<td>2%</td>
</tr>
<tr>
<td>Similar sounding words do not trigger SO</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 5-10: Phonologically similar words

The lesser success of the second task is attributed to the heuristics that were used in the phonological component. Changing the heuristics, described in Section 3.3.1, brings the success rate for the correctly generated wordplay to 70%, but considerably reduces the speed with which
the recognizer finds script overlap and script opposition. The heuristic function that was modified selected the targets that had a higher frequency and familiarity values than the source. In some cases, the jokes recognition goes from minutes to several hours. This is not surprising: the heuristic function was designed to reduce the number of words considered for wordplay. Adding lexical and semantic information into the heuristic function may reduce the speed of the word generator without sacrificing the accuracy of it.

The rest of the analysis of the jokes based on the phonological similarity detection is similar to that of word-ambiguity-based jokes. Table 5-11 shows the number of texts (jokes and non-jokes) correctly identified, when all words in the text were known to the detector.

<table>
<thead>
<tr>
<th></th>
<th>Word Knowledge</th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes</td>
<td>All words known</td>
<td>18%</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>Not all words known</td>
<td>8%</td>
<td>72%</td>
</tr>
<tr>
<td>Non-jokes</td>
<td>All words known</td>
<td>0%</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>Not all words known</td>
<td>2%</td>
<td>78%</td>
</tr>
</tbody>
</table>

Table 5-11: Known words for phonological jokes from *The American Heritage First Dictionary*.

Table 5-12 shows the number of texts that contain word meanings that were actually used in texts, and texts identification as jokes or non-jokes.

<table>
<thead>
<tr>
<th></th>
<th>Meaning Knowledge</th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes</td>
<td>All needed meanings are known</td>
<td>14%</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>Not all needed meanings are known</td>
<td>12%</td>
<td>72%</td>
</tr>
<tr>
<td>Non-jokes</td>
<td>All needed meanings are known</td>
<td>0%</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>Not all needed meanings are known</td>
<td>2%</td>
<td>78%</td>
</tr>
</tbody>
</table>

Table 5-12: Known meanings for phonological jokes from *The American Heritage First Dictionary*. 

107
Table 5-13 shows the number of texts that were identified as jokes or non-jokes and their relationship to meanings of salient words or concepts, known to the computer.

<table>
<thead>
<tr>
<th>Salient Knowledge</th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All salient meanings are known</td>
<td>18%</td>
<td>2%</td>
</tr>
<tr>
<td>Not all salient meanings are known</td>
<td>8%</td>
<td>72%</td>
</tr>
<tr>
<td>Non-jokes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All salient meanings are known</td>
<td>0%</td>
<td>20%</td>
</tr>
<tr>
<td>Not all salient meanings are known</td>
<td>2%</td>
<td>78%</td>
</tr>
</tbody>
</table>

Table 5-13: Salient meanings, needed for the phonological joke detection from *The American Heritage First Dictionary*.

Table 5-14 shows the number of texts that were identified as jokes or non-jokes corresponding with the number of texts with conceptual relationships known by the ontology.

<table>
<thead>
<tr>
<th>Relationship Knowledge</th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All needed relationships are known</td>
<td>2%</td>
<td>0%</td>
</tr>
<tr>
<td>Not all needed relationships are known</td>
<td>24%</td>
<td>74%</td>
</tr>
<tr>
<td>Non-jokes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All needed relationships are known</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Not all needed relationships are known</td>
<td>2%</td>
<td>98%</td>
</tr>
</tbody>
</table>

Table 5-14: Relationships between concepts, needed for phonological joke detection, from *The American Heritage First Dictionary*.

The results look similar to the word-ambiguity-based jokes in that the joke detection rate is small, and the number of known words, meanings, and needed relationships is small. The use of the ontology considerably improves the results. This consistent improvement of the detection with the increase of knowledge is not surprising. A parallel can be made here between people
and computers: when people do not have the necessary background to understand what the joke is about, they do not understand the joke. It seems that computers follow the same trend in joke detection. Thus, all necessary knowledge for joke understanding from a human’s point of view was added to the ontology to test if it would improve the detection results.

Table 5-15 shows the results of the humor detector for phonological jokes when the ontology contains all the necessary knowledge needed to understand texts (jokes or non-jokes), and when the speed of the phonological component is not sacrificed for accuracy.

<table>
<thead>
<tr>
<th></th>
<th>Identified as jokes</th>
<th>Identified as non-jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes</td>
<td>54%</td>
<td>46%</td>
</tr>
<tr>
<td>Non-jokes</td>
<td>12%</td>
<td>88%</td>
</tr>
</tbody>
</table>

Table 5-15: Text detection when all required information is known.

The numbers in the table indicate that computational recognition of phonological jokes is also possible (71% of joke and non-joke recognition), and is better than a chance selection. The non-jokes are detected as such with a higher accuracy than jokes. Taking into account that the detector declares a text to be a joke only when it finds script overlap and script opposition, this is not surprising. The non-jokes should not contain script overlap and script opposition; otherwise, they would be jokes. Whether the ontology contains enough knowledge to detect the meaning of a text or not, the text (understood or not) is a non-joke. The jokes should contain script overlap and script opposition. However, in order for them to be detected, the ontology has to have the necessary knowledge, and the jokes have to be linguistically simple. Consider the following joke as an example:

Text 29: Where do birds invest their money? In the stork market.
In order for this text to be recognized, “stork” has to be converted to “stock”, and the ontology has to know what the “stock market” is. It can be argued that since “stock” is a real word, and stock market is a place where stocks are sold and bought, this joke fits the criteria of allowable jokes. However, “stock market” should be treated as a phrase, rather than two separate words. The mapping to the ontology should be done at the phrase level, not word level. Unless the language processor can handle it (and this one does not), the expected script overlap and script opposition will not be found.

5.3 CAN JOKES BE RECOGNIZED BY COMPARING THEM WITH ALREADY KNOWN JOKES?

To answer this question, 50 previously detected jokes and 50 new jokes are tagged with the knowledge resources from the General Theory of Verbal Humor (GTVH). The theory is used to compare previously used jokes and the new jokes. All 50 new jokes must be similar to the 50 previously detected jokes, according to the GTVH criteria.

According to the GTVH, two jokes differ if at least one of the knowledge resources of the jokes differs. The knowledge resources are: SO, LM, TA, NS, SI, LA. The similarity of jokes is proportional to the number of same knowledge resources.

For this experiment, the jokes are considered similar, if they contain the same SO and the same LM, as tagged by a human. Thus, for each of the 50 known jokes, there is a previously undetected joke that contains the same SO and LM. This means that each previously undetected joke has at least two of the same knowledge resources as a detected joke.

Table 5-16 shows the number of correctly identified jokes.
Correctly identified | Incorrectly identified
--- | ---
Detector | 76% | 24%
Random | 50% | 50%

Table 5-16: Joke detection with previously known jokes

Because jokes are represented using description logic, subsumption may be used to determine if an unknown text is a joke. If an unknown text is subsumed by a known joke, then the unknown text is a joke. The results of this test are shown in Table 5-17.

Correctly identified | Incorrectly identified
--- | ---
Regular detector | 76% | 24%
Subsumption | 8% | 92%

Table 5-17: Subsumption vs. regular joke detector

An interesting question to be answered is: do the results of subsumption depend on joke similarity? This question, however, is considered for future research.

**5.4 CAN JOKES BE RECOGNIZED WHEN AN ONTOLOGY DOES NOT HAVE ALL OF THE REQUIRED BACKGROUND KNOWLEDGE TO PROCESS THE MEANING OF TEXT?**

To answer the question, two classes of recognized jokes are compared. The difference between the classes is whether the ontology contains all information that is needed to fully understand a joke, or only partial information. The results are shown in Table 5-18.

<table>
<thead>
<tr>
<th>Ontological Information</th>
<th>Number of recognized jokes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete ontological information</td>
<td>57%</td>
</tr>
<tr>
<td>Incomplete ontological information</td>
<td>12%</td>
</tr>
</tbody>
</table>

Table 5-18: Detection of texts according to their themes

The numbers in the table indicate that complete ontological information is very helpful for joke detection. However, these numbers tell only part of the story. It was suspected that
incomplete ontological information is useful for jokes that have multiple words that can trigger classification. Thus, it is important to know how many such jokes there are. And, of these jokes, how many have been detected with incomplete ontological information? Moreover, out of those that have been detected with complete ontological information, how many could be detected with an incomplete one? The jokes that are used in this dissertation do not contain a sufficiently high number of this joke type to answer these questions. These questions will be answered in future research.

5.5 Are Some Jokes Easier to Recognize Than Others?

The five jokes themes, compared with respect to their detection success, is shown in Table 5-19.

<table>
<thead>
<tr>
<th></th>
<th>Fairytale</th>
<th>Monster</th>
<th>Mammal</th>
<th>Non-mammal animal</th>
<th>People</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jokes identified as such</td>
<td>14</td>
<td>15</td>
<td>11</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>Jokes identified as non-jokes</td>
<td>6</td>
<td>5</td>
<td>9</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Non-jokes identified as such</td>
<td>18</td>
<td>19</td>
<td>17</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>Non-jokes identified as jokes</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5-19: Detection of texts according to their themes

Looking at the table, it can be seen that the detection success is different for different categories. For example, Monster jokes seem to be much easier to recognize than Mammal jokes. A possible explanation is that Monster jokes used in the dissertation are unintentionally similar, while the Mammal and People jokes have more dissimilarity between jokes in the same theme.

However, this difference is not statistically significant: F=0.79, which is less than the critical value of F(4, 120). Thus, the null hypothesis of *there is no difference in the joke recognition
between categories cannot be rejected. On the other hand, there is a statistically significant difference between recognition of jokes and non-jokes as such: \( F=14.69 \), which is greater than the critical value of \( F(1, 190) \). Thus, if we are interested in the null hypothesis of there is no difference in the correct detection of joke and non-jokes, it should be rejected.

The results indicate that computational detection of these jokes is possible. This is interesting not only because it is possible to detect a small subset of humor, but also because the classification of humorous or non-humorous texts is explainable in semantic terms. In other words, we are no longer looking at associations between features of texts and their classification. Instead, we are looking at cause and effect structure. The cause is the necessary and sufficient conditions for a text to be humorous, as described in Script-based Semantic Theory of Humor. The effect is the classification of a text as a joke.
6 Conclusion

The purpose of this work was to answer whether or not computational detection of humorous texts is possible, when the information required for text processing is computationally accessible. The domain of short children jokes was selected to narrow the required information. The domain was further restricted to jokes that are based on semantic ambiguity or phonological similarity.

For a humor detector to be feasible, an algorithmic representation of a humor theory is needed. In this work, the Script-based Semantic Theory of Humor was used as a model for the algorithms. The theory was formalized to be suitable for computational needs. It should be noted that the same formalization can be used not only for jokes that are based on semantic ambiguity or phonological similarity, but verbally-expressed jokes in general.

There are several motivations for doing computational humor in general. They range from more-user friendly computers and human-computer interaction to applications that specifically use humor for education, advertising and error detection. None of these applications is likely to be successful without sophisticated natural language processing software, that is capable of detecting an intended meaning of a non-humorous text, and reasoning about information in this text. Unless a humor detector is built on top of all of the above, it remains a toy system, unusable in real world. But what would happen if such sophisticated natural language processing software were already implemented? Would it be possible to recognize humor, or is humor completely undetectable by the machines? It is this last question that this work addresses. To simulate software that can reason about information in the text, a description logic ontology
was created. The texts are semi-manually mapped into the information in the ontology. Using information that is in the ontology, the question of humor detection is answered.

The humor detector consists of three components, representing spelling, pronunciation and meaning of words. These components correspond to human level of processing natural language text. The component that deals with the meaning of words consists of: a description logic knowledge base, containing general information about the world in its TBox, and newly (dynamically) added information from the text in its ABox; a semantic role labeler, that uses information in the ontology to find relationships between words in texts; and a humor analyzer that uses one of the humor theories to determine if a text, as represented in the ABox of the description logic knowledge base, is humorous.

The central question that this dissertation aimed to address was whether humor recognition of natural language texts is possible when the knowledge needed to understand the texts is available in a machine-understandable form (in this case, description logic knowledge base). In order to understand better what kind of humor is possible to detect, and what kind of knowledge is needed to detect it, the following questions were answered:

• Is it possible to recognize jokes that are based on lexical ambiguity?
• Is it possible to recognize jokes that are based on words with similar pronunciation?
• Can jokes be recognized by comparing them to already known jokes?
• Can jokes be recognized when an ontology does not have all of the required background knowledge to process the meaning of a text?
• Are some types of jokes easier to recognize than others?

The results indicate that computational detection of both kinds of jokes that were considered is possible (all with the same degree of difficulty). Moreover, it is only possible when the salient
information needed to understand jokes is available in the ontology. This result is interesting not only because it is feasible to detect a small subset of humor, but also because the classification of humorous or non-humorous texts is explainable in semantic terms. In other words, we are no longer looking at associations between features of texts and their classification. Instead, we are looking at a cause and effect structure. The cause is the necessary and sufficient conditions for a text to be humorous, as described in the Script-based Semantic Theory of Humor. The effect is the classification of the text as a joke.

It has always been known that there are certain patterns in jokes. Formal semantic theories of humor try to capture such patterns. A computational model is the ultimate formalization: if the information can be presented in the machine-understandable way and software can be written for the computer to do something compatible with a theory, the theory passes the final test. So the most significant insight is that a computational system can be designed to confirm, to a certain acceptable degree of reliability, that the Script-based Semantic Theory of Humor is sound because most of the jokes that the theory recognizes as such are correctly detected by the system.

It must be obvious, however, that this research effort had to be largely exploratory in nature. The sample was small. The ontology was created manually for the task. The percentage of success could be higher in certain cases. The comprehensive sophisticated meaning-based natural language processing software was only emulated. The next challenge is to use the real-life resources of the ontological semantic technology, with its 10,000-concept ontology, 120,000-entry lexicon, and a fully functional OntoParser, and to reconstruct the model on its basis. The corpus of jokes and non-jokes must be augmented and the methodology of its selection brought up to the state-of-art level of experimental design. One has a strong intuition that the results of such an enhanced non-toy detector will be much improved, but it will take a significant future
research effort to support this belief. The first step, the research effort reported here, has been certainly encouraging.
7 BIBLIOGRAPHY


Semantic Role Labeling Demo. (n.d.).


APPENDICES

APPENDIX A: COST TABLE (HEMPELMANN, 2003)

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
<th>Column 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
</tr>
</tbody>
</table>

130
APPENDIX B: HIGH LEVEL JOKE ALGORITHM AND EXAMPLE

With an example of “A man walks into a bar. Ouch”

1. Read all words from text.
2. Map (autonomously) each word from the text into ontological concepts.
   \[
   \begin{align*}
   \text{man} & \rightarrow \text{Man}_1w; \\
   \text{walks} & \rightarrow \text{Enter}_1w; \\
   \text{walks} & \rightarrow \text{Hit}_1w; \\
   \text{walks} & \rightarrow \text{Walk}_1w; \\
   \text{bar} & \rightarrow \text{Room}_1w; \\
   \text{bar} & \rightarrow \text{Pole}_1w; \\
   \text{ouch} & \rightarrow \text{Pain}_1w.
   \end{align*}
   \]
   a. For some words, concepts or relationships don’t exists
      For example, \text{a}, does not have concepts
   b. Give a user an option to supply a concept for such words
      No thanks
   c. If the user chooses to do so, accept this mapping
   d. For each word that corresponds to chosen concepts, create instance if it does not
      already exist in the ontology. If a word has two meanings (i.e., corresponds to 2
      concepts) create one instance for a union of these concepts.
      \[
      \begin{align*}
      \text{instance man of concept Man}_1w, \\
      \text{instance walk of concept Hit}_1w & \text{ OR Enter}_1w \text{ OR Walk}_1w,
      \end{align*}
      \]
   e. Create a list of all word instances in the text, call it List\_D, and a list of concepts,
      call it List\_C such that each member of List\_D is an instance of some concept in
      List\_C.
3. Annotate semantic relationships between instances in List\_D
   a. Find all parents of all concepts in List\_C, create a list of parents, List\_P
      for each \text{i} in List\_P{
         for each \text{j} in List\_P{
            if there exist a relationship \text{R} between \text{i} and \text{j}{
               find an instance of subclass of \text{i} that is in List\_C, call it \text{subject}
               find an instance of subclass of \text{j} that is in List\_C, call it \text{object}
               add (\text{subject, R, object}) to list List\_R
            }
         }
      }
      List relationships in List\_R:
      \[
      \begin{align*}
      0 & (\text{walk, tr_theme_r1, man}) \\
      1 & (\text{walk, tr_agent_r1, man}) \\
      2 & (\text{walk, tr_effect_r1, walk}) \\
      3 & (\text{man, tr_agent_of_r1, walk}) \\
      4 & (\text{walk, tr_origin_r1, bar}) \\
      5 & (\text{walk, tr_location_r1, bar}) \\
      6 & (\text{walk, tr_theme_r1, bar}) \\
      7 & (\text{walk, tr_destination_r1, bar}) \\
      8 & (\text{walk, tr_path_r1, bar}) \\
      9 & (\text{walk, tr.instrument_r1, bar}) \\
      10 & (\text{walk, tr_effect_r1, ouch}) \\
      11 & (\text{man, tr_agent_of_r1, ouch})
      \end{align*}
      \]
b. Give a user an option to remove relationships that are incorrect
   8, 6, 4, 2, 0

c. Give a user an option to add relationships that are missing
   No thanks

d. Record removals, additions
   0 (walk, tr_agent_r1, man)
   1 (man, tr_agent_of_r1, walk)
   2 (walk, tr_destination_r1, bar)
   3 (walk, tr_instrument_r1, bar)
   4 (walk, tr_effect_r1, ouch)
   5 (man, tr_agent_of_r1, ouch)

4. Add this information to the ontology. Now we can reason about specific information contained in the text.

5. Find scripts that correspond to the text: If concept in List_C is a subclass of Event_1w, it is a script

   Events Enter_1w, Hit_1w, Walk_1w:
   Enter_1w=Move_1w AND tr_destination_r1.Place_1w
          tr_effect_e1.Positive_feeling_1
          tr_agent_r1.Animal_1
   Hit_1w=Move_1w
          tr_instrument_r1.Physical_object_1
          tr_agent_r1.Animal_1
          tr_effect_r1.Hurt_1
          tr_theme_r1.Physical_object_1
   Walk_1w=Move_1w
          tr_instrument_r1.Foot_1w
          tr_agent_r1.Animal_1
          tr_effect_r1.Exercise_1 OR Location_change_1

6. If at least two scripts are found:
   a. Find script overlap for each pair of scripts (Enter/Hit, Hit/Walk, Walk/Enter).
      i. At the very least, it is all OR concepts from the joke (the overlap comes from the same spelling)
         Hit_1w or Enter_1w or Walk_1w
         Restaurant_1w or Pole_1w
      ii. Additionally, it is the concepts that are in both scripts of a pair.
         Man_1w in Enter_1w/Hit_1w
         Man_1w in Hit_1w/Walk_1w
         Man_1w in Walk_1w/Enter_1w
   b. Find script opposition for each pair of scripts
      i. Convert OR concepts into pairs of AND concepts. Add each pair to the KB and check KB consistency.
         Add Hit_1w AND Enter_1w
         Add Hit_1w AND Walk_1w
         Add Enter_1w and Walk_1w
         Add Restaurant_1w and Pole_1w
      ii. Find effects of each pair of scripts, check if disjoint
         Positive_feeling_1, Hurt_1w are disjoint
         Positive_feeling_1, Exercise_1 OR Location_change_1 are not disjoint
Hurt_1w, Exercise_1 OR Location_change_1 are not disjoint

c. If there is a pair of scripts that overlap and oppose, return JOKE
Overlap exists, opposition exist in Hit_1w/Enter_1w -- JOKE

d. If not – go to 7

7. Add phonological component to find similar sounding words. Find familiarity and frequency of all words in the text.

8. Starting from last word, for each word that familiarity and frequency < medium familiarity and frequency of all words in the text
   a. Look for similar sounding words. Similarity is defined in terms of cost of removing, adding or replacing each phoneme of a word
   b. Report the list of the 100 best entries.
   c. In this list consider only those words whose familiarity and frequency is greater than that of the word that is being replacing
      i. For each such word or until joke is found, repeat steps 2-6c.
      ii. If 6c does not report JOKE result, and the list is exhausted, the text is NOT a JOKE.
APPENDIX C: JOKES USED IN THE DISSERTATION

1. What do you call a witch that climbs up walls? Ivy!
2. Why did the witches go on strike? They wanted sweeping reforms!
3. How do witches on broomsticks drink their tea? Out of flying saucers!
4. Ali Baba didn't know it but there were four women locked in the cave with all that jewelry. What were their names? Ruby, Jade, Coral and Pearl!
5. When is a piece of wood like a king? When it's a ruler!
6. Why is a well-attended prince like a book? Because he has so many pages
7. What is the favorite subject of young witches at school? Spelling
8. What did the doctor say to the witch in hospital? With any lick you'll soon be well enough to get up for a spell.
9. What should you expect if you drop in on a witch home unexpectedly? Pot luck!
10. Optician: "Have your eyes ever been checked?" Ogre: "No, they've always been red!"
11. What do you call two witches who share a broom stick? Broom mates!
12. What do you call a wizard from outer space? A flying sorcerer!
13. What did Robin say when he nearly got hit at the archery contest? "That was an arrow escape!"
14. Why do dragons sleep during the day? So they can fight knights
15. What do witches ring for in a hotel? Broom service
16. How do witches keep their hair out of place? With scare spray
17. Why do witches only ride their brooms after dark? That's the time to go to sweep!
18. How do witches tell the time? By looking at their witch watch
19. What do you call a wizard who lies on the floor? Matt
20. When does a prince get very wet? When he becomes the reigning monarch
21. When a teacher closes his eyes, why should it remind him of an empty classroom? Because there are no pupils to see
22. Why did the teacher put the lights on? Because the class was so dim
23. Did they play tennis in ancient Egypt? Yes, the bible tells how Joseph served in Pharoah's court
24. What do history teachers use when they want to get together? Dates
25. Did you hear about cross eyed teacher? He couldn't control his pupils
26. Why did the teacher wear sunglasses? Because the students were so bright
27. Why was the math book sad? It had too many problems
28. Doctor, Doctor, I feel like a pack of cards. I'll deal with you later
29. Doctor, Doctor I've broke my arm in two places. Well don't go back there again then
30. Were you long in the hospital? No, I was the same size that I am now!
31. Why were the early days of history called the dark ages? Because there were knights
32. Did the Native Americans hunt bear? Not in the winter!
33. Where did knights learn to kill dragons? At knight school!
34. Doctor, Doctor my husband smells like fish. Poor sole!
35. When do astronauts eat? At launch time!
36. How do bees get to school? By school buzz
37. When were King Arthur's army too tired to fight? When they had lots of sleepless knights
38. What was Camelot famous for? Its knight life
39. Are you choking? No, I really did
40. Doctor, Doctor, some days I feel like a tee-pee and other days I feel like a wig-wam. You're too tents.
41. Dr Frankenstein: How can I stop that monster charging? Igor: Why not take away his credit card?
42. What do you do with a green monster? Put it in the sun until it ripens!
43. 1st monster: I was in the zoo last week. 2nd monster: Really? Which cage were you in?
44. What happened at the cannibal's wedding party? They toasted the bride and groom
45. What do you call a dog owned by Dracula? A blood hound!
46. Why did the cannibal live on his own? He was fed up with other people
47. How do you make a werewolf stew? Keep him waiting for two hours
48. What sort of soup do skeletons like? One with plenty of body
49. What happened when the ghost asked for a whiskey at his local bar? The bartender said: Sorry sir, we don’t serve spirits here
50. How do you know that you are talking to an undertaker? By his grave manner
51. Why is six scared of seven? Because seven eight nine
52. What is a vampire's favorite soup? Scream of tomato
53. What kind of medicine does Dracula take for a cold? Coffin medicine
54. What did the shy pebble monster say? I wish I was a little boulder
55. What does a vampire stand on after taking a shower? A bat mat
56. What do ghouls do when they are in hospital? They talk about their apparitions
57. How do ghosts keep fit? By regular exorcise
58. What is the best way to get rid of a demon? Exorcise a lot
59. What do you call a demon who slurps his food? A goblin
60. What do vampires have at eleven o’clock every day? A coffin break
61. What are the cleverest bees? Spelling bees!
62. Why is a sofa like a roast chicken? Because they're both full of stuffing
63. What is the insect's favorite game? Cricket
64. What is life like for a wood worm? Boring
65. Why was the glow worm unhappy? Because her children weren't that bright
66. What do you call a bird that’s been eaten by a cat? A swallow
67. Why did the perch sit on the fish? The fish was a perch
68. Where do little fishes go every morning? To school!
69. What kind of fish is useful in freezing weather? Skate
70. What do you get if you cross a trout with an apartment? A flat fish
71. When is the best time to buy parakeets? When they're going cheap
72. What birds spend all their time on their knees? Birds of prey
73. What do you call a bunch of chickens playing hide-and-seek? Fowl play
74. What do you call a bird that lives underground? A mynah bird
75. What is a parrot's favorite game? Hide and Speak
76. What do bees do if they want to use public transport? Wait at a buzz stop
77. Where do birds invest their money? In the stork market
78. Why are mosquitoes religious? They prey on you!
79. Why didn't the two worms get on Noah's Ark in an apple? Because everyone had to go on in pairs
80. Which fish can perform operations? A Sturgeon
81. Why do cats chase birds? For a lark
82. What's worse than raining cats and dogs? Hailing taxi cabs
83. Why is it called a "litter" of puppies? Because they mess up the whole house
84. What happens to a dog that keeps eating bites off of the table? He gets splinters in his mouth
85. Why aren’t leopards any good at hide-and-seek? They are always spotted
86. Why did the farmer call his pig 'Ink'? Because he kept running out of the pen
87. What do you get from pampered cow? Spoiled milk
88. What's big and grey with horns? An elephant marching band
89. Why did the elephant eat the candle? For light refreshment
90. Why were the elephants thrown out of the swimming pool? Because they couldn't hold their trunks up
91. What is a French cat's favorite pudding? Chocolate mousse
92. What does the lion say to his friends before they go out hunting for food? Let us prey
93. Why did the cat frown when she passed the hen house? Because she heard fowl language
94. What happens when it rains cats and dogs? You can step in a poodle
95. What do you get if you cross a computer and a Rottweiler? A computer with a lot of bites
96. How does a lion greet the other animals in the field? Pleased to eat you
97. Why should you be careful when you are playing against a team of big cats? They might be cheetahs
98. What cats drink on hot summer afternoons? Miced tea
99. Why didn’t they let the wildcat into school? They knew he was a cheetah
100. What did the farmer call the cow that would not give him any milk? An udder failure!