COMPUTERIZED DETECTION OF SEMANTIC EQUIVALENCE AMONG SENTENCES IN NATURAL LANGUAGE

By

Celibette Michelle Ossorio Laracuente

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS

December, 2011

Approved by:

Bienvenido Vélez Rivera, Ph.D Member, Graduate Committee	Date
Manuel Rodríguez Martínez, Ph.D Member, Graduate Committee	Date
José Fernando Vega Riveros, Ph.D President, Graduate Committee	Date
Betsy Morales Caro, Ph.D. Graduate Studies Representative	Date
Pedro I. Rivera Vega, Ph.D Chairperson of the Department	Date

Abstract of Dissertation Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of Master of Science

COMPUTERIZED DETECTION OF SEMANTIC EQUIVALENCE AMONG SENTENCES IN NATURAL LANGUAGE

By

Celibette Michelle Ossorio Laracuente

December 2011

Chair: José Fernando Vega-Riveros
Major Department: Electrical and Computer Engineering

In natural language, different sentences can express the same meaning, or a sentence can be modified without altering its meaning. This is called "semantic equivalence". Semantic equivalence can be resolved humans, but it is still an unresolved problem for computerized systems. Shallow semantics is used in this research to recognize semantic equivalence in sentences in English, which makes the approach domain-independent. A case-based system is developed, which uses the Stanford Natural Language parser to obtain grammatical information and looks for patterns identified in each one of the cases. A modified version of the Microsoft Research Paraphrase corpus (MSRP) with 451 sentence pairs was utilized to test the system. An average rate of 89.80% of successful equivalence detection was obtained, which compares favorably with the success rate between 63.94% to 74.00% reported in the literature. The main contributions of this thesis are the development of a system to determine semantic equivalence among sentences using shallow semantics, the identification of the particular structures of the dependencies generated by the parser

for each case and the development of a corpus of semantic equivalence sentences.

Resumen de Disertación Presentado a Escuela Graduada
de la Universidad de Puerto Rico como requisito parcial de los
Requerimientos para el grado de Maestría en Ciencias

DETECCIÓN COMPUTARIZADA DE EQUIVALENCIA SEMÁNTICA ENTRE ORACIONES EN LENGUAJE NATURAL

Por

Celibette Michelle Ossorio Laracuente

Diciembre 2011

Consejero: José Fernando Vega-Riveros
Departamento: Ingeniería Eléctrica y Computadoras

Las oraciones en lenguaje natural pueden expresar el mismo significado o una oración puede ser modificada sin alterar su significado. A esto se le llama "equivalencia semántica". La equivalencia semántica puede ser resuelta por un ser humano, pero todavía este problema no ha sido resuelto para los sistemas computarizados. En esta tesis desarrollamos un sistema que usa semántica superficial para reconocer equivalencia semántica en pares de oraciones escritos en el idioma inglés, lo cual hace que la aproximación sea independiente del dominio. El sistema desarrollado para esta tesis consiste de un algoritmo basado en casos, el cual usa el Analizador de Lenguaje Natural de Stanford para obtener la información gramatical de las oraciones y observar los patrones identificados en cada uno de los casos. Una versión modificada del corpus de Investigación de Paráfrasis de Microsoft (MSRP por sus siglas en inglés), la cual tiene un total de 451 pares de oraciones, fue usada para probar el sistema. Se obtuvo una tasa promedio de 89.80% de detección exitosa de equivalencia, la cual compara favorablemente con la tasa de éxito de entre 63.94% a 74.00% reportado en la literatura. Las principales contribuciones de ésta tesis son

el desarrollo de un sistema para determinar equivalencia semántica entre oraciones usando semántica superficial, la identificación de estructuras particulares de las dependencias generadas por el analizador para cada caso y el desarrollo de un corpus de oraciones de equivalencia semántica.

This thesis is dedicated to God and my family.

ACKNOWLEDGMENTS

First I am sincerely grateful to my advisor, Dr. José Fernando Vega-Riveros, for trusting me and giving me his guidance, knowledge and support to develop this thesis. I also thank the members of my Graduate Committee: Drs. Manuel Rodríguez Martínez and Bienvenido Vélez Rivera.

Thank to my boyfriend, José Javier Rodríguez, for his unconditional support and love all this time, and for always being there, especially when I needed him the most. Many thanks to my family for their support as well.

My most sincere thanks to Dr. Nayda G. Santiago Santiago, for treating me like a daughter. Thanks for all the support, lunches, advice and scolding which were a great help to me. Many thanks to my academic counselor, Mrs. Sandra Montalvo Solorzano for her assistance and friendship during my graduate studies. Thanks to the professors that advised me at certain times, Dr. Isidoro Couvertier and others, and thanks to the friends that support me during this process.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

Figure page

CHAPTER 1
INTRODUCTION

Today the World Wide Web has become one of the most important sources of information. The documents that are on the web are mostly written in Hypertext Markup Language (HTML) [2], a language created to display text, images, and links, and where links redirect to other web pages [3]. The main objective of HTML is to show information to the user and allow him or her to navigate through information seamlessly. The major part of the current documents and information found on the web was designed to be read by a human, not to be manipulated meaningfully by machines because computers cannot process the semantics of web content reliably [4]. This approach has a problem; it is designed to show the person what the web page has but not to help the person find what he or she is looking for, especially if the person is not an expert. With time, many users have acquired the necessary expertise to find the information that they are looking for but it takes time to develop this expertise. What happens to inexperienced users? The language that they use (natural language) is not the same as the language used by computers to store information. The queries to retrieve this information are based on keywords. A more natural way to communicate with computers would be throught natural language.

Natural language is the way that humans communicate. It has two forms: written and spoken. In written language, and as spoken language, humans communicate using words, which are the basic units of sentences [5], since they make up phrases, which in turn make up sentences. Sentences are the largest unit of grammar [6] and

consist of two main parts: the subject and the predicate [6]. Appendix A includes a more detailed explanation of Grammar.

Natural language researchers have spent a great effort trying to solve the problem of understanding natural language. Researchers have identified three tasks for understanding natural language: grammar, semantics, and parsing. Grammar defines the structure of natural language [7], while semantics is defined as the study of meaning in language [8]. Meaning refers to "the sense of a linguistic expression, sometimes understood in contrast to its referent" [9]. In Artificial Intelligence and the Semantic Web, it is a common practice to use semantic nets, schemas, ontologies and taxonomies, among others to represent knowledge and semantics as concepts and their relationships. Parsing finds the function of each word in a sentence, aiming to get the appropriate meaning [7].

Natural language depends mostly on context, which is very difficult for computers to manage [10]. In human communication, one of the problems that context can solve is ambiguities. Another problem in natural language is that an idea can be expressed with different words and sentences. The problem here is that "semantic equivalence", or many forms exist to refer to the same thing. Computers cannot identify if two sentences are semantically equivalent, which means that the two sentences have the same meaning. The work concentrates on designing an approach to find semantically equivalent sentences.

1.1 Background and motivation

Semantics can be integrated in the World Wide Web, in order to create a more intelligent web. In fact, Tim Berners-Lee created the World Wide Web, with the purpose of including semantics, but this goal has not been achieved yet. Since the

beginning of the web, Berners-Lee has tried to find ways to transform the current Web into a type of web which he called "semantic web". He believed that the transformation of the current web into a semantic web would be straightforward, but today such a concept has not been achieved [11]. Semantic web, as defined by Tim Berners-Lee, "is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [4]. It means that computers can "understand" what a user search exactly means. Semantic Web, also called Web 3.0, "will enable machines to comprehend semantic documents and data, not human speech and writings" [4]. The Semantic Web is a web of data expressed in some suitable language created for representing human natural language, i.e., to handle what a user means when he or she performs a web search. Semantic theory provides an account of "meaning" in which the logical connection of terms establishes interoperability between systems [11]. As the name implies, semantic web is a web based on semantics, natural language semantics.

Some research fields on semantics in natural language processing are used in semantic web. Semantic equivalence and semantic entailment are some of them. Semantic entailment is defined as the deduction of information of one sentence from information transmitted in a previous sentence [12], while semantic equivalence occurs when two sentences have the same meaning [12]. Semantic entailment and semantic equivalence have a strong relationship [12] [13] and sometimes entailment is used to detect semantic equivalence [13].

1.2 Problem Statement

In natural language, a person can express the same meaning using different expressions [14]. Also a sentence or phrase can be modified without altering its

meaning [15]. This is called "semantic equivalence". Semantic equivalence can be resolved by a human, but it is still an unresolved problem for computerized systems. The problem addressed in this thesis is that of a computerized system recognizing semantic equivalence in sentence pairs written in natural language.

1.3 Scope of the work

In this work, an approach and tests to determine semantic equivalence in pairs of sentences in the English natural language, was developed. Sentence pairs can be entered through the console of the system or can be read from a text file. Semantic equivalence will be determined using a set of semantic equivalence cases defined in this work. The set of cases uses shallow semantics for several reasons:

1. The approach is domain-independent;
2. There is no need to have ontologies or any other form of knowledge representation; and
3. The system does not need to compute a logic representation of the sentences.

Knowledge-independence can be approached in at least two ways:

- Using syntactic information; or
- Using complex machine learning algorithms that can gradually learn and construct knowledge-representations from texts.

Using ontologies or any other form of knowledge representation would make the system domain-dependent or would require an extremely large knowledge base.

Computing a logic representation of sentences is also an unsolved complex problem in AI which involves multiple forms of logics, e.g., description logic, modal logic, temporal logic, among others.

In this thesis we will approach the problem of determining semantic equivalence independent of knowledge domain using syntactic information.

1.4 Significance of the study

Since the semantic equivalence is an understudied topic in natural language processing, a methodology to determine English semantic equivalence between sentences pairs in natural language is proposed and developed. A system like this can help in different areas and applications such as the search of similar topics, search of similar documents, summarization of documents, and question-answering systems, among others. The semantic web research is another area that can benefit from a system like the one proposed in this thesis.

1.5 Overview of Methodology

In this work, semantic equivalence between sentences was determined developing a series of semantic equivalent cases which were defined selecting different common forms to represent sentences. Shallow semantic techniques were used for this work. To obtain the grammatical information of the sentences, the Stanford Natural Language parser was used. The algorithms in this thesis use the typed dependencies collapsed obtained from the parser, the WordNet lexical database, and some CSV files. WordNet was used to verify synonyms and antonyms and the Comma Separated Value (CSV) files simulated a database of verbs, pronouns, and contractions.

1.6 Contributions

Our main research contributions are:

- The development of a system which determines semantic equivalence among English sentences based on 14 semantic equivalence cases of shallow semantics.
- The identification of the particular structure of the dependencies generated by the Stanford Natural Language Parser for each case (template dependencies), and the development of the algorithm using these dependencies based on the patterns observed in each of the studied cases.
- The development of a corpus of semantic equivalence sentences.

1.7 Organization of thesis

This thesis is divided into six chapters. The second chapter reviews literature related to this research work, from natural language parsing, discussing different aspects of computational semantics, as well as ways to analyze related research work. The third chapter discusses the parser grammar definitions, including a review and a comparison of the different types of parser dependencies, a review of semantics in general and semantic equivalence in particular. This is followed by a discussion of the theory of each one of the semantic cases studied, and their mathematical form, including the combined cases. The fourth chapter describes the methods and the tools used to develop the semantic case algorithms, including the Stanford Natural Language parser, WordNet, the Java language peculiarities, and the library tool that manages CSV files. It also includes the design approach used to develop each semantic case, a discussion in detail of each case, and examples. The fifth chapter presents and describes the corpus used to test the system, present the tests applied to measure the performance of our system and analyze the results obtained from these tests, measuring the false positive and false negative rates, and comparing the system with similar ones. The sixth chapter presents the conclusions and the suggested future work.

CHAPTER 2
LITERATURE REVIEW

In the following chapter, the literature review relevant to this thesis is presented. This chapter presents topics about natural language processing, semantics, semantic web, applications of semantic in semantic web, and other systems.

2.1 Natural Language Parser

In Natural Language Processing, NLP, one of the manners to analyze a sentence is with the use of a parser. A parser is a tool that works out the grammatical structure of the sentences [16]. Two types of parsers exist: the statistical, also known as stochastic or probabilistic [17], and the deterministic, also known as rule-based parser [1].

In a deterministic parser the next step is always known; it is predictable. These parsers have the peculiarity that next state will be determined by the current state. They have a finite number of rules which are predetermined. For each rule, there is a specific rule to follow. These parsers are good in programming languages where the grammar is defined and usually do not change.

On the other hand, in a statistical parser, the next step is determined using probabilities. If the next state cannot be predicted exactly, the parser needs to be stochastic. For each rule, there are a number of rules with a probability in which the rule with the highest probability is chosen. These parsers are good in natural

language where the grammar is in constant evolution, and a fixed number of rules are not suitable. As in natural language, the number of rules is very large, and it is not likely to have enough rules. The most used approximation is the probabilistic parsers.

Russell and Norvig defined grammar as "a collection of rules that defines a language as a set of allowable strings of words" [1]. Chomsky Normal Form is a grammar format in which the rules are in a specific format:

$$X \Rightarrow YZ$$

or

$$X \Rightarrow \text{word}$$

[1] [18] [19]. The Chomsky normal form is one of the simplest and most useful forms of context-free grammars [18]. The CYK (Cocke-Younger-Kasami) algorithm requires its grammar in a Chomsky Normal Form [1].

The CYK algorithm calculates the probability of the most probable tree, instead of analyzing all the parse trees. This algorithm is the best one for context-free grammars [1]. Context-free grammars (CFG) are an effective method for describing languages, especially those which have a recursive structure [18]. These grammars were first used in the study of natural language, and can capture important aspects of the relationships between nouns, verbs, and prepositions with their respective phrases [18]. CFGs are commonly used in the grammars of natural language and programming languages [1] [18]. Chomsky was the first who explored CFGs in the natural language context [20].

A probabilistic context-free grammar, PCFG, is a context-free grammar but with the addition that every string has a probability assigned [1]. A PCFG has many rules and each one has a probability, therefore, learning grammar from a Treebank is better than using a knowledge engineering approach. The fact that PCFGs learn grammar from a Treebank shows us that the PCFGs are trainable. A Treebank is a "corpus of correctly parsed sentences" [1], where each sentence in the corpus has a syntactic structure added to it [21]. The Penn Treebank is the most commonly known and consists of 3 million words annotated with parts of speech tags and parse-tree structure [1].

A lexicalized PCFG is a type of PCFG where the probabilities of the rules not only depend on the closeness of words in a sentence, but on the relationship between words in the parse tree as well [1]. These relationships are done with the help of the head of the phrase. The head of a phrase is the most important word in the phrase and is used to get the relationships between words [1]. The Lexicalized PCFG uses the relations in the parse tree to decide what rule to apply.

There are some statistical parsers, including the Stanford Natural Language Parser, Minipar and Link. The Stanford Natural Language Parser was developed by The Stanford Natural Language Processing Group [21]. This parser generates a Treebank structure while the other two do not [21]. In addition, for each input sentence, the Stanford Parser generates a Treebank parse tree [21]. The Link parser, instead of generating a parse tree, generates a linkage [21]. This parser uses a link grammar [21] unlike the Stanford parser which uses a Dependency grammar [17].

The Stanford parser generates parses with high accuracy. In addition, this parser is most accurate and robust when trained on very large corpora when compared to the Link and Minipar parsers [17]. Direct comparison of these three parsers is complicated. First, sometimes the parsers do not agree on the dependents of the relationships or on the type of relationship as such. The three parsers differ on the collapsed dependencies [17]. The three parsers are dependency parsers [17].

The Stanford parser does not perform very well on questions, while Minipar is confused by punctuation and conjunction, and Link obtains incorrectly dependencies on the MX relation and it has problems with conjunctions also [17].

In the Stanford Natural Language Parser, the structure of sentences can be represented using typed dependencies and phrase structures [17]. Typed dependencies are a representation of the dependencies that exist between individual words, while labeling them with grammatical relationship names [22] [17]. They are organized as triples [22] [21] , which are defined [21] as a relationship between the subject, the object, and the predicate of the sentence. The predicate corresponds to the relationship between the subject and object. In the Stanford Dependency Parser, the grammatical relationships are hierarchically organized with the most generic relationship as the root [22] [17]. On the other hand, the phrase structures are representations of phrases nested as multi-word components [17].

2.2 Semantic Web and applications

Semantic Web, also called Web 3.0, is a more intelligent web in which information is given well-defined meaning. It "will enable machines to comprehend semantic documents and data" [4]. The semantic web goal is that computers "understand" what a user search exactly means. The Semantic Web represents the evolution of a

web that consisted primary of documents created to be read by a human to a web in which the information is easily understandable and manipulated by machines [11]. The vast majority of the documents in the current web are written in HTML [2], and their main objective is to show information to the user and allow him or her to navigate through information seamlessly. The current documents and information found on the web was designed to be read by a human, not to be manipulated meaningfully by machines [4]. This approach has the problem that it was designed to show the person what the web page has, but not to help the person find what he or she is looking for.

In order for the semantic web to work, the information must be structured in collections, and must have access to the inference rules that perform the automatic reasoning [4]. The power of the Semantic Web will be gained when programs that gather content of different sources of the web, manage the information and exchange the results [4].

In the Semantic Web, the semantic search is similar to a question and answer system (Q&A system), as Watson [23] , in that the user poses queries with an intended context, not just keywords or isolated words. In contrast to a Q&A system, in semantic search and retrieval, the system does not look for a specific answer, but for relevant information, much like a librarian would do. The problem with semantic queries is that they can be sentences or parts of sentences which can be expressed in many diverse forms including semantically equivalent sentences.

Semantic web technologies can be used in a wide spectrum of fields. In biology and bioinformatics, they are used in combination with ontologies, to facilitate the investigators research by storing and retrieving relevant information, such as

genome-scale molecular information, genes information to detect diseases or disease-related processes [24] [25].

The education field, specifically "e-learning", has been another application of the semantic web. In [26] they propose a framework for educational systems that use reasoning rules over distributed annotations for the generation of user targeted hypertext relations in a dynamical way . Another e-Learning tool called CHESt is a database of educational multimedia clips in which students can enter a question and the system searches into the educational clips, giving back the clip or clips containing possible answers [27]. CHESt uses a semantic search mechanism that transforms the question entered in a RDF query format in order to have the same structure in both the question and the database [27].

Another possible application of semantic web in education is in information repositories, such as academic repositories. An example is the system described in [28] which integrates relevant information from different data sources about academic people and their expertise. The problem they want to solve is to join in a database all the academic information of academic experts, in order to facilitate finding relevant information about other academicians, including research interests, publications, websites, etc. They argue that creating a database manually with the needed information is too expensive and maintaining the information updated is a problem [28].

Xin, et al, build an application in which semantics is used to create conference calendars. Basically, these calendars use semantic annotations to extract the desired conference information from the web. The problem they try to solve is to enter the information manually in a calendar which can lead to some disadvantages including

that the information presented may be much more than what is of interest to the user. An example of this issue occurs when a user wants to see information about certain specific conferences; a traditional system would show all the conferences and the user would have to search through them to find what he or she wants. The different configuration alternatives that are available in this calendar solve this issue. With an application such as the semantic conference calendar, a user specifies the name of the conference he or she is interested in and the system looks for the details of it and add them to the calendar [29].

Semantic Web also can be used in image information retrieval. Semantic annotations combined with ontologies can be used to retrieve all semantic information from images and helps to find better image descriptions. It can be useful to make better image searches [30].

Lexical resources such as encyclopedias assume the reader possesses a large amount of common-sense knowledge and therefore, offer limited or no information about word meaning, which makes disambiguation a very difficult or sometimes impossible task to achieve. Explicit Semantic Analysis is a method that can be used to resolve these issues due to its ability to address synonymy and polysemy which are two of the most important problems in NLP [31].

NLP deals with the problem of managing the natural language semantics. Given that in natural language the same information can be addressed in different forms, the variability of natural language is a major problem [12].

2.2.1 Ontologies

Ontologies are collections of information which are a basic component of the semantic web [4], and are fundamental technologies for knowledge representation

of the semantic web [4] [32]. According to [2], an ontology "is a formally defined explicit specification of a shared conceptualization". In artificial intelligence, ontologies were created to reuse and share knowledge [2] [32]. In ontologies there exist individuals, classes, and properties. An individual represents an object, while a class represents a group of individuals, and the properties represent the relationships between individuals or the relationships between individuals and data values [33] [32]. An individual can have different names, but this problem can be solved with the use of an Universal Resource Identifier (URI), which is assigned to each individual [32]. The function of a URI is to identify a resource. "Associating a URI with a resource means that anyone can link to it, refer to it, or retrieve a representation of it" [11]. An equivalence problem can occur when different URIs may refer to the same individual at the same time [32].

2.3 Shallow Semantics and Deep Semantics

Semantics can be studied using two approaches: shallow semantics and deep semantics. In the case of reasoning, which is closely related to semantics, Joseph Giarratano and Gary Riley [34] refer to shallow reasoning as the reasoning that usually uses a single rule or a few inferences and is based on experience, while they refer to deep reasoning as deep knowledge, which implies "a deep understanding of the subject". Deep reasoning may require longer chains of inference associated with an understanding of the subject in an abstract sense [34]. Based on this definition, we can define shallow semantics as the kind of semantics based only on grammatical information without resorting to causal chains or any other abstract knowledge. The shallow semantic information of sentences can be identified by semantic parsers [35]. Shallow semantics can cover a wide variety of subject domains without requiring specialized knowledge representation skills. With shallow semantics, problems such as ambiguities, that need more specific semantic knowledge, cannot be resolved. In

the same way, we can define deep semantics as the kind of semantics which needs in-depth knowledge of the subject. Decades of research in natural language processing demonstrate that the use of deep structural, relational and semantic properties of text is a necessary step towards supporting higher level tasks [15]. Deep semantics understands the abstract sense of the particular subject. Ontologies are an example of deep semantics. Deep semantics cover a narrow variety of subject domains but with specialized knowledge of the domains.

2.4 Semantic Similarity

Another important term in natural language, related to semantic equivalence is semantic similarity. Similar sentences are those which express the same meaning sometimes with different words or are topic related [12]. Due to the variability of natural language, the similarity in sentences is not simple to identify, but the interest in research of sentence similarity is growing up and having an important role in natural language processing [12] [36].

2.4.1 Semantic Entailment and Semantic Equivalence

In natural language semantics, semantic entailment and semantic equivalence are important terms. There is a strong relation between semantic entailment and semantic equivalence [12] [13].

Semantic entailment is defined as the deduction of information of one sentence from information transmitted in a previous sentence or sentences [12], i.e., two texts are entailed if the meaning of one of them can be inferred from the meaning of the other [37] [15] [12] [14] [5] [38]. The entailment between texts is closely related to the entailment between words [5]. Semantic entailment is a rapidly evolving research

area in NLP [37] and a fundamental [15] and complex [37] problem in natural language understanding and processing. So far, there is no appropriate solution for textual entailment determination [15] [38]. All linguistic processing methods can be included in a textual entailment system or a semantic equivalence system [38]. But, many of the existing systems use logical equivalence to attack the problem of recognizing semantic equivalence, translating text to theorems which is also a very complex problem. Other systems use Machine Translation to detect semantic equivalence of two sentences in different languages; if the translation of one is found in the other, they are considered semantically equivalent. Some of the PASCAL RTE Workshop applications use lexical matching to address the problem, while another combines lexical matching with WordNet similarity measures achieving better results than the ones which do not use WordNet [13]. On the other hand, in many other studies, WordNet and other lexical resource bases are also used to measure lexical entailment [5] [14]. Another approach used for entailment in other systems is the syntactic parse trees similarity of the texts, while other systems used logic provers and some of them increase the provers using world knowledge axioms [14].

Entailment recognition requires lexical, syntactic and semantic processing [38]. According to [5], there are four manners to recognize textual entailment. The first one derives linguistic information from the pair of texts, and casts the inference recognition as a classification problem. The second one uses conditional probability (bag of words). The third one expresses the knowledge from the pair of texts in some representation language that can be associated with an inferential mechanism. The last approach is based on the classical AI definition of entailment. They build models of the world in which the two texts are respectively true, and then check whether the models associated with one text are included in the models associated

with the other text [5].

Seeing it from another point of view, the mentioned approaches can be summarized in two main categories: one based on knowledge techniques and the other based on machine learning and statistical methods [5].

Sometimes, entailment is used to detect semantic equivalence. Although it is not exactly the same as semantic equivalence, they are related [13]. Semantic equivalence deals with the problem of multiple phrases or sentences having the same meaning. However, research in this area is scarce. One example of semantic equivalence is the paraphrase of sentences [12]. An example of paraphrase sentence is "Knowledge management is important in modern companies" and "In modern companies, is very important the management of knowledge". Also, it can be said that two sentences are semantically equivalent if they share the same information [12].

Given that in natural language the same information can be addressed in different forms, the variability of natural language is a major problem. Inference itself is uncertain and has a probabilistic nature [5]. When entailment occurs in both directions, it is said that there is the semantic equivalence, called paraphrase [38].

A semantic equivalence system can benefit greatly other research areas [13] by matching knowledge expressed in different forms. There exist some approaches to try to recognize semantic equivalence. Patterns are used in Information Extraction to identify semantic equivalence on semantically-equivalent texts [13].

2.5 Related Work

Both semantic equivalence and semantic entailment have many applications in natural language processing and Semantic Web.

One application is depurating summaries [13]. One of the problems that the multi-document summarization research faces is that many times the most important information of a document is repeated over and over, causing the summary to contain the same information repeated in various sentences. This problem triggers the interest to detect semantic equivalence to eliminate repetition in texts to improve the quality of the summary [13]. The system proposed by [13], tries to detect semantic equivalence in texts using an entailment approach. The system uses a decision tree classifier composed by lexical, semantic and grammatical information. The system uses WordNet, VerbOcean, and Latent Semantic Indexing tools to determine entailment relationships. With WordNet, the system identifies if synonyms are used. The most important feature of this system is that they look for the longest common subsequence on the sentence pairs to detect if contradictions exist in the pair. They examine the verb semantics of the subsequences to find synonyms, near-synonyms, negation or antonyms in the pair [13].

In this system, they use some semantic equivalence features, as the ROUGE metrics and the related Cosine similarity measure. The ROUGE metrics compares a summary done by a human expert with a summary done by a computer, assessing the quality of the last one. When synonyms are widely used or sentence structure is modified (compositional paraphrase), entailment is difficult to distinguish. As the compositional paraphrase, the syntactic paraphrase can be difficult to distinguish in this system, but with the use of ROUGE metrics, this type of paraphrase can be identified [13]. An example of syntactic paraphrase is the pair sentence: “I drive the

car" and "the car was driven by me". Both sentences have the same meaning but are written in different manners.

To perform the system tests, the RTE corpus and the Microsoft Research Paraphrase corpus (MSRP) [39] were used [13]. The RTE corpus was developed for the Pascal RTE Challenge [13] [38]. The latter is a set of manually matched sentences obtained from different sources, and classified in seven categories: Comparable Documents (CD), Machine Translation (MT), Information Retrieval (IR), Information Extraction (IE), Question Answering (QA), Reading Comprehension (RC), and Paraphrasing (PP) [13] [37] [38] [5]. This corpus consists of two training development sets, which have 287 and 280 sentence-pairs, and one test set, which contains 800 sentence-pairs [13] [38]. The sentence-pairs have two components called "text" and "hypothesis". The corpus sentences come from different news datasets and corpora pertaining to the different NLP tasks where textual entailment is used [13] [38]. On the other hand, the MSRP corpus is a set of pairs of paraphrases which were constructed manually [37] and contains 5,801 sentences pairs obtained from thousands of internet sources like news sites, where 4,076 were training pairs and 1,725 were test pairs [13] [12]. This corpus was built by a process in which each sentence pair was examined by two human judges who established, in binary form, if one sentence of the pair is entailed by the other [12]. One sentence is considered entailed by the other if it is close in meaning, paraphrases or "semantically equivalent". If the two judges did not agree in the result of a sentence pair, a third judge was called to evaluate the pair. Finally, the judges determined that only 3900 (67%) of the original 5801 pairs were "semantically equivalent" [37] [40].

The MSRP corpus was used in two ways: without modification and modified. Newman et al [13] modified the MSRP corpus to create another one where they

obtained an equal number of positive and negative instances. The evaluation metric used to rate the system performance was looking for the output of the sentence pairs and classify them as correct or incorrect. They obtain a higher performance in the modified version of the corpus than the original one [13].

As the authors mention in [13], a system for detecting semantic equivalence may also work in other areas, such as a question answering systems. If a question is rewritten as a statement, the system can try to find semantic equivalence between the statement and statements in the document or documents where the search is being done, and produce the answer to the question originally made [13].

As mentioned before, there is a strong relationship between semantic entailment and semantic equivalence [13] [12]. Some of the current systems use semantic entailment trying to approximate semantic equivalence, translating the text to theorems, while other systems use techniques used in Machine Translation. Basically the approach is that when someone translates from language A to language B, and another translation is found in language B, semantic equivalence between both translations could be assumed. The problem of textual entailment recognition can be simplified into a graph matching problem using parse trees [13]. In the system created by [13] a decision tree classifier is used, along with WordNet, to identify entailment between pairs of sentences.

In [37] shallow semantics is used in a knowledge system that evaluate and recognize semantic equivalence between two sentences. They used the WordNet relations to measure the semantic equivalence of two sentences. By shallow semantics they mean a combination of basic syntactic matching between the partial predicate-argument structures with a thesaurus-based semantic equivalence. The

semantic equivalence was obtained by the relationship between the representation of the meaning of a sentence and the lexical relatedness. The first is known as the partial Predicate Argument Structure, PAS, and the second uses the WordNet distance. The system uses the results of the parser to build the partial Predicate Argument Structure, and then the generated constituents are compared one by one. The similarity of partial syntactic structures of the sentences is the first equivalence marker that the system uses. The simple shallow partial predicate argument structure only uses the verb, with its subject and object, if it exists. The system uses two parsers, and can set them in two ways: set one of them as the default and use the second only when the first one does not produce a parse, or set both parsers with equal priority and let the system choose for the parser that produces more PAS for the sentence. The system is not able to handle compound synonyms, i.e., synonyms that have more than one word, either multiword or independent words which can be taken as with one single term. They used Microsoft Paraphrase corpus and the data of the PASCAL RTE challenge corpus to evaluate the performance of their system [37]. However, it is not clear whether they compute semantic equivalence or entailment since they used both terms interchangeably. Apparently, they implement a system that uses shallow semantics to recognize semantic equivalence between two sentences, but the paper title is about entailment and they mention that their system is to evaluate textual entailment.

In [12], they want to measure semantic similarity between sentences using the structure of sentences, particularly verb argument-structures. The method takes into account the semantic structure of the sentences to avoid the problem of "semantic loss" which occurs when the syntactic construction of the sentences is ignored when converting the sentences into a "bag of words". Equivalent sentences "should share similar verb-argument structures". The similarity between sentences was measured

using a formula that calculates the similarity score. WordNet was used to calculate the similarity score of the verb structures [12]. For this system, four evaluation metrics were defined: recall, precision, F1, and accuracy. Recall refers to the proportion of correctly predicted similar sentences compared to all the similar sentences. Precision is defined as a "proportion of correctly predicted similar sentences compared to all predicted similar sentences", F1 as a "uniform harmonic mean of precision and recall", and accuracy is a "proportion of all correctly predicted sentences compared to all sentences" [12].

To perform the system tests, Achananuparp et al used the Microsoft Research Paraphrase corpus (MSRP), and the third PASCAL recognizing textual entailment challenge (RTE3) data set [12]. In the RTE3 each sentence pair has two text segments referred to as "text" and "hypothesis". If the "hypothesis" can be entailed by the "text", the sentence pair can be considered equivalent [12]. It is worth noting that the RTE3 corpus used by [12] is different to that used in [13] because RTE3 is an improved version.

Based on these results, Achananuparp et al concluded that structural approach is better for computing the similarity of highly asymmetric sentences, such as the RTE3 pairs, than those with similar length [12].

Identifying semantic equivalence or semantic entailment in a sentence pair needs a deeper understanding of the meaning of sentences. According to the results obtained, they claimed deeper semantic measures can recognize the same or greater number of positive pairs than those using a vector space approach. This is more evident in entailment [12]. The method proposed by [36] takes into account both

semantic information and word order of sentences to measure the sentence similarity.

Traditional techniques for detecting similarity between documents focus on shared words. The problem with these techniques is that the similar long texts usually contain many words that are shared between them but short texts only share a few of the words or none. In the semantic similarity measures, we have to take into account both semantic and syntactic information because the two contribute to the sentence meaning, but, generally, in current methods only one of them is considered.

In the method proposed by [36], the word similarity weight is used to get the semantic similarity, while the syntactic similarity is obtained using the similarity of word orders. The overall similarity in sentences is obtained with the combination of the two. There is a great deal of research on measuring semantic similarity among two texts, but not between two sentences. Some methods used to measure semantic similarity between sentences are modifications of methods used to measure semantic similarity between long texts. Because these methods were created for long texts, these methods are inefficient, require human input, and cannot be adapted to some application domains [36].

The method presented in [36] measures word similarity using a formula that uses the distances of the paths between words in WordNet to obtain a score which determines the similarity between the given words [36]. The most uncommon words give more information than more common. This is known as word significance. The semantic similarity among two sentences is calculated using the sum of the word similarity and the information content of the words of sentences [36].

The syntactic similarity among sentences is calculated assigning an index number, which corresponds to the position of the word in a sentence. They create a vector containing the index numbers of the words, one vector for each sentence. Calculating the correlation coefficient between the two vectors, the syntactic similarity measure is obtained. The overall sentence similarity between two sentences is calculated adding the semantic similarity and syntactic similarity of both sentences using a smoothing factor. To perform their system tests they used the Microsoft paraphrase corpus, MSRP. Their results were evaluated in both forms: graphically and using a formula. In the graphical way, the results were shown evaluating false positives vs. true positives, while with the formula they used four formulas: accuracy, recall, precision and F-measure [36].

De Salvo Braz et al. [15] describe a system which used machine learning and models inference as an approach to determine semantic entailment between two texts. Their approach consists of a knowledge representation, a knowledge base, and a subsumption relationship. The knowledge representation is based on description logic and is used to "represent the surface level text, augmented with induced syntactic and semantic parses and word and phrase level abstractions" [15]. The knowledge base consists of semantic and syntactic rules which describe the subsumption relationship between the left hand side (body) and the right hand side (head) of the rule. They used an extended subsumption algorithm to determine the subsumption between the two representations of each rule [15]. The augmented representation encodes numerous possible representations using rewrite rules that allow the sentence representations to be modified. Many rules have a large number of heads, for example synonyms and different forms to name a specific person (John F. Kennedy, JFK) [15].

Their approach to semantic entailment is mainly based on a hierarchical representation of natural language sentences defined on concept graphs. The language they used is called Extended Feature Description Logic (EFDL), which is a Description-Logic inspired language [15].

In the concept graphs, the nodes represent elements, word or phrases, the attributes of the nodes represent the properties of the elements, and the labeled edges between two nodes represent the relationships between the elements. The algorithm that [15] used is a sound algorithm for semantic entailment and used WordNet. Their system was tested using the PASCAL challenge data set (RTE) and got 84.00% and 87.5% for the QA and MT subtasks [15].

In [5], WordNet is used to get information to be applied to a probabilistic setting used for RTE to measure lexical entailment. Their system was tested using RTE1 and RTE2 datasets. The RTE2 corpus, similarly to its predecessor the RTE1 corpus, was developed to test the second PASCAL RTE challenge and consist of 800 annotated development pairs and 800 annotated test pairs [5]. In [14], the semantic knowledge of texts is extracted using semantic axioms. Their semantic approach for RTE is based on logic, with the premise that a given text entails another if the meaning of the first logically implies the meaning of the second. In order to get the meaning of both texts, the semantic relationships must be identified. If the relationships between texts are similar, entailment can be established.

This chapter presented the literature related to this research work, from natural language parsing, discussing different aspects of computational semantics, and the way to analyze related research work. In the following chapter the theoretical

framework of this thesis is presented.

CHAPTER 3
THEORETICAL FRAMEWORK

In this thesis we work with semantic equivalence among pairs of sentences in natural language. For the development of this work some knowledge and some tools to manage the equivalence in the sentence pairs is needed. Grammar is used to extract a semantic representation of a sentence using a parser. With the information provided by the parser, we have developed a method to find semantic equivalence among two sentences. See Appendix A, for a brief summary of English grammar that is essential to the understanding of the work in this thesis.

3.1 Parser Grammar Definitions

In the Stanford Natural Language Parser 55 grammatical relations, also known as typed dependencies, have been defined to represent the relationships between words in a sentence. These relationships are described in the "Stanford Typed Dependencies Manual" [41]. A grammatical relationship is a triplet composed of the name of the relationship, the governor and the dependent [22][17][41]. A triplet in a sentence is defined by [21] , as a relationship between subject, object, and predicate of the sentence, being the predicate the relationship and the subject and object the parameters. The governor is the superior term in the dependency connection, while the dependent is the inferior term. Dependency connections are the dependency relationships between the words in a sentence. A dependent, also known as subordinate, is the complement of the governor. These definitions of governor and dependent were originally defined by Lucien Tesnière in "Éléments de syntaxe

structurale" [42]. He defined the Dependency Grammar, which the Stanford parser implemented. The parser grammatical relationships are standard grammatical relations. The most generic grammatical relationship generated by the Stanford parser is *dependent* or *dep*. It is used when no other relationship between two words is found by the parser [22][17][41].

3.1.1 Parser dependencies

As previously stated, in the Stanford Natural Language Parser there are 55 grammatical relationships. These relationships are classified into five categories of typed dependencies:

- Basic dependencies
- Collapsed dependencies
- Collapsed dependencies with propagation of conjunct dependencies
- Collapsed dependencies preserving a tree structure
- Non- Collapsed- dependencies

In the basic typed dependencies, each word in a sentence is dependent of another word. This type of dependency uses all defined dependencies that form a tree structure.

In the collapsed typed dependencies, there are more dependencies considered than the basic ones. It uses also the non-projected dependencies and the dependencies that break the tree structure and convert it into a directed graph. This type of dependency collapses the basic dependencies, making fewer, simpler and more descriptive dependencies. The dependencies that become collapsed are those which contain prepositions, conjunctions, and information about the referent of relative

clauses. Usually, the collapsed part of the dependencies, from which two dependencies become collapsed, turns into part of the relationship name, *e.g.*, two dependencies *prep(based-7, in-8)* and *pobj(in-8, LA-9)* will become collapsed into one single dependency *prep_in(based-7, LA-9)*.

The prepositions the parser collapses can be viewed on Table 1 and Table 2 of the "Stanford Typed Dependencies Manual" [41]. The conjunctions the parser propagates can also be viewed in Table s of the referred manual.

The collapsed typed dependencies with propagation of conjunct dependencies are an extension of the collapsed typed dependencies and for that reason, a tree structure is not guaranteed. Propagation of conjuncts occurs when a conjunction exists, the relationships of the first conjunct is propagated to the second conjunct. For example, in the sentence "Bell, a company which is based in LA, makes and distributes computer products.", there exists a conjunction between the verbs "makes" and "distributes", which are known as conjuncts, and this conjunction is propagated adding two dependencies to the collapsed representation. The subject and object relationship of "makes" will be propagated to "distributes" creating the dependencies: nsubj(distributes-13, Bell-1) and dobj(distributes-13, products-15).

The collapsed typed dependencies preserving a tree structure include all the collapsed typed dependencies except those that break the tree structure. The collapsed typed dependencies with propagation of conjunct dependencies are not included in these types of dependencies because these dependencies do not guarantee a tree structure.

The non-collapsed typed dependencies include all the basic dependencies and all other that do not do any kind of collapsing or propagation of conjuncts.

Using the example sentence: “Bell, a company which is based in LA, makes and distributes computer products.”, the following table shows the typed dependencies produced by the parser:

Table 3–1: Typed dependencies produced by the parser

Basic dependencies	Collapsed dependencies	Collapsed with propagation of conjunct dependencies	Collapsed dependencies preserving a tree structure	Non-collapsed dependencies
nsubj(makes-11, Bell-1)	nsubj(makes-11, Bell-1)	nsubj(makes-11, Bell-1)	nsubj(makes-11, Bell-1)	nsubj(makes-11, Bell-1)
det(company-4, a-3)	det(company-4, a-3)	nsubj(distributes-13, Bell-1)	det(company-4, a-3)	det(company-4, a-3)
appos(Bell-1, company-4)	appos(Bell-1, company-4)	det(company-4, a-3)	appos(Bell-1, company-4)	appos(Bell-1, company-4)
nsubjpass(based-7, which-5)	nsubjpass(based-7, company-4)	appos(Bell-1, company-4)	nsubjpass(based-7, which-5)	nsubjpass(based-7, which-5)
auxpass(based-7, is-6)	auxpass(based-7, is-6)	nsubjpass(based-7, company-4)	auxpass(based-7, is-6)	auxpass(based-7, is-6)
rcmod(company-4, based-7)	rcmod(company-4, based-7)	auxpass(based-7, is-6)	rcmod(company-4, based-7)	rcmod(company-4, based-7)
prep(based-7, in-8)	prep_in(based-7, LA-9)	rcmod(company-4, based-7)	prep_in(based-7, LA-9)	prep(based-7, in-8)
pobj(in-8, LA-9)	conj_and(makes-11, distributes-13)	prep_in(based-7, LA-9)	conj_and(makes-11, distributes-13)	pobj(in-8, LA-9)
cc(makes-11, and-12)	nn(products-15, computer-14)	conj_and(makes-11, distributes-13)	nn(products-15, computer-14)	cc(makes-11, and-12)
conj(makes-11, distributes-13)	dobj(makes-11, products-15)	nn(products-15, computer-14)	dobj(makes-11, products-15)	conj(makes-11, distributes-13)
nn(products-15, computer-14)		dobj(makes-11, products-15)		nn(products-15, computer-14)
dobj(makes-11, products-15)		dobj(distributes-13, products-15)		dobj(makes-11, products-15)
				ref(company-4, which-5)

3.2 Semantics

In natural language, semantics is defined as the study of meaning in language [8]. The referent of a linguistic expression is one thing or one person in the world [8]. As referent, reference and sense are important terms in semantics. Reference is what the speaker refers to, or is talking about, i.e., the object, while sense is not entirely a thing, rather is an abstraction [8]. Sense can be explained as the correct understanding of what the speaker is talking about. When a person understands correctly what a speaker is talking about, it is said that the person gets the correct sense. The most used example to understand this is "the morning star" and "the evening star". Both phrases have the same reference, both refer to the planet Venus, but both have different senses, "the morning star" is the brightest star which appears sometimes before sunrise, and the "evening star" is the brightest star which appears sometimes after sunset. Another example is the sentences: "I will go to the theater to see the play "Edipus Rex"." and "I will take the children to play at the park tomorrow." In these two sentences the word play has two different senses. In the first sentence, the word "play" refers to a theather drama, while in the second sentence, the word "play", is a verb that denotes the action of engaging in sports or recreation.

Meaning can be divided into two parts: speaker meaning, and sentence or word meaning. Speaker meaning refers to the intention of the speaker, what he or she means when using language. Sentence meaning refers to the meaning of a sentence itself, and has to do with the language used. The same applies to word meaning [8].

In artificial intelligence, the term semantics refers to the representation and sharing of knowledge between machines [33].

We will explore one type of semantic relationship between pairs of sentences, specifically semantic equivalence. This is an understudied field in natural language processing. The equivalence between two sentences is determined if the sentences share the same meaning. Semantic relationship between two sentences can be explored using entailment or paraphrasing [12]. Entailment is defined by Achananuparp et al. [12] as the deduction of information of one sentence from information transmitted in a previous sentence, and is defined by De Salvo Braz et al. [15], as the task of determining if one sentence entails another. If the meaning of a sentence can be inferred from the meaning of another sentence, it is said that the second sentence entails the first one [37][14][5][12][15]. Determining entailment between two sentences is complex work that does not have a complete solution as of today, but research is rapidly evolving [37]. Entailment is difficult to recognize if synonyms are used or if the structure of a sentence has been changed [13]. Some researchers use WordNet as a helping tool to determine textual entailment relationships [5][37].

According to Russell & Norvig [1], two sentences are logically equivalent if the first sentence entails the second and the second entails the first. This equivalence can be represented in a mathematical form:

$$A \iff B \text{ iff } (A \models B) \ \wedge \ (B \models A)$$

Based on this definition, it can be established that two sentences are semantically equivalent by a similar argument.

A representation of entailment is shown in Figure 3–1. This figure illustrates the relationship between entailment and semantics showing the connections between text and the real world.

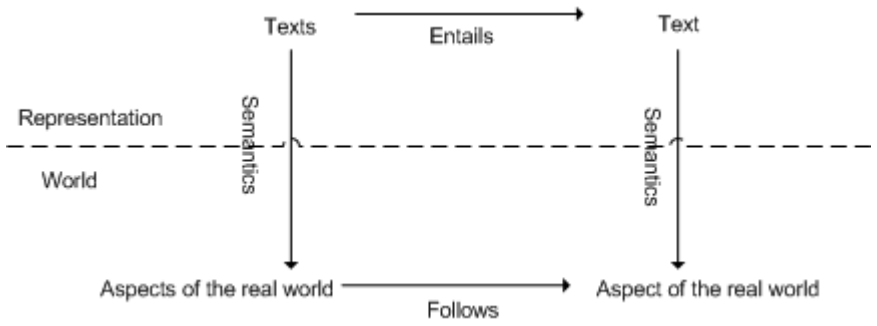

Figure 3–1: Graphic representation of Entailment. This figure is an adaptation of Figure 7.6 of [1] (Reproduced with permission of the author)

Figure 3–3 shows a graphical representation of entailment between two sentences A and B. Entailment occurs either, when information in sentence B follows from information transmitted by sentence A, but not the other way around, or the information in sentence A follows from information transmitted by sentence B, but not the other way around. On the other hand, figure 3–2 shows a graphical representation of equivalence between two sentences A and B. Equivalence requires that information in sentence B follow from information transmitted by sentence A and, at the same time, information in sentence A follow from information transmitted by sentence B.

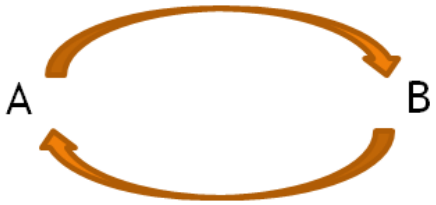

Figure 3–2: Graphic representation of Semantic Equivalence.

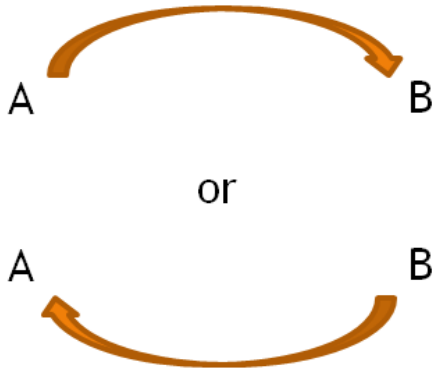

Figure 3–3: Graphic representation of Semantic Entailment.

3.3 Semantic Equivalence

A sentence is composed of phrases, and phrases are composed of words. Sentences have grammar and semantics. In this work we assume that all sentences are grammatically correct and we analyze if two sentences, written in English, are semantically equivalent.

Equivalence in sentences, as previously stated, is determined if the sentences share the same meaning [12]. Two sentences are equivalent if they have the same meaning and sense. Many times, meaning depends on sense because sense can disambiguate the meaning of a sentence. A sentence may have more than one meaning when read alone, but sense gives the sentence the real meaning. Sense is related to context. For example the sentence "The chicken is ready to eat." may have two interpretations. One is that the chicken itself is going to eat and the other is that the chicken is ready to be eaten.

Semantics can be studied using two approaches: shallow semantics and deep semantics. Shallow semantics can be defined as the kind of semantics based only on grammatical information without resorting to causal chains or any other abstract knowledge [34]. Using grammatical information, certain patterns can be observed,

from which we can determine grammatically how to find equivalence in a given case. Shallow semantics can cover a wide variety of subject domains but do not require specialized knowledge of them. Problems that need more specific semantic knowledge, such as ambiguities, cannot be resolved using shallow semantics. This type of problem can be resolved using the deep semantics approach. Deep semantics can be defined as the kind of semantics which needs in-depth knowledge of the subject. Deep semantics covers a narrow variety of subject domains but requires specialized knowledge.

Figure 3–4 shows graphical representations of shallow and deep semantics. Taking the rectangle as a representation of knowledge, it is seen that shallow semantics can cover a wide variety of subject domains with no specialized knowledge, while deep semantics can cover one or a few subject domains but with specialized and deep knowledge.

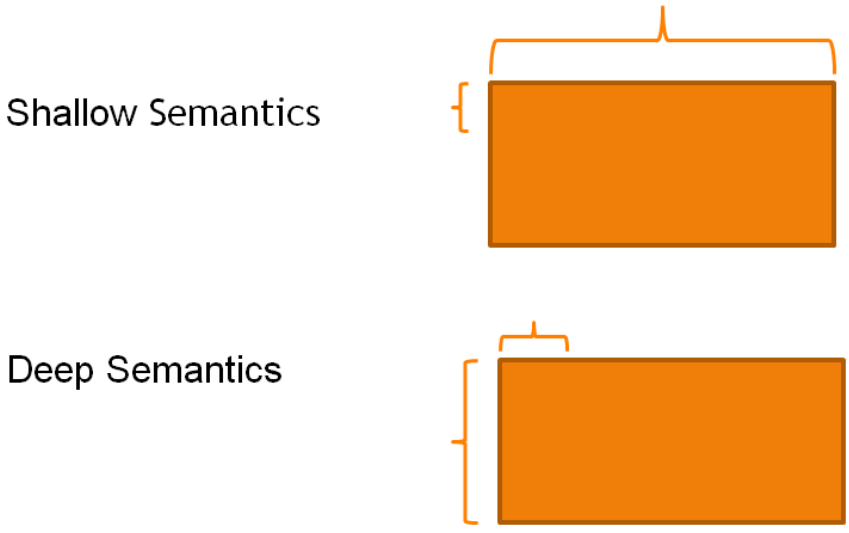

Figure 3–4: Graphic representation of Shallow Semantics vs. Deep Semantics.

In this work, different cases where two sentences are semantically equivalent were analyzed. These cases were stated from some common forms of writing two

sentences conveying the same meaning but grammatically different.

Using shallow semantics, a series of cases were defined and are described below.

- Case 1 - Identical Sentences
- Case 2 - Sentences Using Synonyms
- Case 3 - Sentences Using Contractions
- Case 4 - Sentences in active and passive voice
- Case 5 - Sentences in Simple Future
- Case 6 - Sentences in Future in the Past
- Case 7 - Sentences using “can” and “be able to” verb forms
- Case 8 - Negation with antonyms
- Case 9 - Combined case: Sentences in active and passive voice using synonyms
- Case 10 - Combined case: Sentences in Simple Future using synonyms
- Case 11 - Combined case: Sentences in Future in the Past using synonyms
- Case 12 - Combined case: Sentences using “can” and “be able to” verb forms using synonyms
- Case 13 - Combined case: Negation of antonym using synonyms
- Case 14 - Combined case: Sentences in Simple Future using contractions

For Cases 1 – 3, the equivalence rules apply both to the entire sentence or to part of it. From Case 4 onwards, we must apply them to the entire sentence. To determine equivalence from Case 4 onwards, we require entering into grammatical analysis, for which we use the Stanford Natural Language Parser. We will use the dependencies between words that represent the grammatical relationships between the words in a sentence. In the following subsections, we will state the semantic

equivalence rules for each case listed above.

3.3.1 Case 1 - Identical Sentences

This case is the most basic one. In this case, the semantic equivalence is defined between two identical sentences. For a pair of sentences to belong to this case, the sentences must have exactly the same words, in the same order. If two sentences are grammatically identical they must be semantically equivalent.

This case may be represented mathematically in the following form:

$$S_1 \iff S_2 \text{ iff } \forall_i(\alpha_i = \beta_i)$$

Where:

$$S_1 = \sum_{i=1}^{n} \alpha_i$$
$$S_2 = \sum_{i=1}^{n} \beta_i$$

α_i represents the words of the first sentence and β_i represents the words of the second sentence.

3.3.2 Case 2 - Identical Sentences using synonyms

In this case, the semantic equivalence is defined for two identical sentences except for the use of one or more synonyms. Synonyms must be single words, because our system does not manage compound synonyms. The system uses WordNet, which is a lexical database developed at Princeton University. Therefore, only synonyms in WordNet are recognized. In WordNet, nouns, verbs, adjectives and adverbs are organized in sets of cognitive synonyms, or Synsets [43]. This researc uses WordNet 3.0. This version of the lexical database has a total of 117,659 synsets. The nouns

category is the more extensive one. A synset is also known as collocation [43].

A pair of sentences belongs to this case if they have the same number of words, and at least one synonym. If two sentences are grammatically identical, except for the use of synonyms, they have the same sense, and so, they are semantically equivalent.

Case 1 is a particular instance of Case 2 because one word can be considered a synonym of itself. Case 1 is handled separately for efficiency avoiding the use of WordNet.

Case 2 may be represented mathematically in the following form:

$$S_1 \iff S_2 \text{ iff } \forall_i(\alpha_i = \beta_i) \vee \exists_i \exists_y(\alpha_i \in S_{y_j} \wedge \beta_i \in S_{y_j})$$

Where:

$$S_1 = \sum_{i=1}^{n} \alpha_i$$
$$S_2 = \sum_{i=1}^{n} \beta_i$$

α_i represents the first sentence words and β_i represents the second sentence words.
S_y is the synonym set
WordNet $= \cup_{j=1}^{n} S_{y_j}$
Let S_{y_j} a synset, where synset is a word set with the same sense j.

3.3.3 Case 3 - Identical Sentences using contractions

In this case, the semantic equivalence is defined between two identical sentences except for the use of contractions. A contraction is a shortened way to represent one or more words. Contractions may appear one or more times in a sentence, in the same or different words. This case has the restriction that the only contractions that it can recognize are those that are in a contractions file used by the system. Currently, the contractions we are managing are those for the following auxiliary verbs: am, is, are, has, have, will, would, not, can, and shall. Note that the contraction of the word "not" is recognized by the system using this file. Any auxiliary verb which does not has a contraction is implicitly recognized by the system when the contraction of its negation is used. Examples of these verbs are "do", "does", "must", among others. Something similar happens with the auxiliary verb "have", which is included in the contractions file. If any word has a contraction that does not appear in the contractions file, the system will not recognize it as a contraction and the result will be incorrect, giving a false negative.

This case may be represented mathematically in the following form:

$$S_1 \iff S_2 \text{ iff } \forall_i((\alpha_i = \beta_i) \vee \exists_j(T_{c_j}(\alpha_i, \beta_i) \vee T_{c_j}(\beta_i, \alpha_i)))$$

Where:

$$S_1 = \sum_{i=1}^{n} \alpha_i$$

$$S_2 = \sum_{i=1}^{n} \beta_i$$

α_i represents the first sentence words and β_i represents the second sentence words. T_c is the contraction table

$T_c = \cup_{j=1}^{n} T_{c_i}$

Let T_{c_i} the tuple formed by one word and its contraction.

3.3.4 Case 4 - Sentences in active and passive voice (Active vs. Passive sentences)

In this case, semantic equivalence is defined between two sentences: one in active voice and the other in passive voice. A pair of sentences belongs to this case if one of the sentences is in active voice and the other in passive voice.

In this case, the sentences in the pair may have a different number of words: passive sentences are longer than active ones. In addition, some transformations occur from active to passive sentences: the order of words change, the verbs are conjugated differently, and if a pronoun is used, it may be conjugated also. From the analysis of the corpus, the verb transforms from active to passive, which means, it changes from simple present or simple past to past participle with the preceding auxiliary verb "to be" in the same tense as the verb in the active sentence. In addition, the preposition "by" is added after the verb in the passive voice. Another transformation that exists is on personal pronouns, if they exist as the subject of the active voice sentence. In this case, the personal pronoun is transformed from subjective to objective. From the passive to active sentence, the transformations are reversed.

These transformations are carried out on the dependencies the parser produces. A nominal subject dependency (nsubj) and a direct object dependency (dobj) always appear in an active sentence, while a passive nominal subject dependency (nsubjpass), a passive auxiliary dependency (auxpass), and an agent (agent) dependency always appear in the passive voice. For the active sentence, the nominal

subject dependency is transformed into an agent dependency in the passive voice, while the governor and dependent also have a transformation. The governors in these dependencies are the verbs in the sentences which show the transformation expressed before, and the dependency dependents are the noun or pronoun in the sentences. In the case the dependents are pronouns, they are transformed too, as stated before; if they are nouns (proper or not) they remain the same. In the same way, the direct object dependency of the active sentence is transformed into a passive nominal subject. The governors of this dependency, which are verbs, suffer the same transformation as the previous dependencies, while the dependents remain the same. In addition to these dependency transformations, the passive voice sentence includes another dependency that is not included in the active voice: the passive auxiliary (auxpass). This dependency is generated by the auxiliary verb existing in the passive voice. As in the active voice it is not an auxiliary verb, this dependency is not generated in active sentences. These mentioned dependencies are required in the sentences in order for them to belong to this case. Both kinds of sentences may have other dependencies, but at least those specified above must exist.

In some cases, sentences in the passive voice do not have the preposition "by" followed by the agent. One example of this case is the pair of sentences:

"We tested the samples."
"The samples were tested."

In this example it cannot be asserted that the subject conducting the tests is the same that was addressed in the active sentence, unless the context is clearly specified. This causes an ambiguity that cannot always be resolved. This type of

sentence pairs is not analyzed in this thesis.

Figure 3–5 shows the transformations occurred when a sentence is transformed from active to passive voice and viceversa.

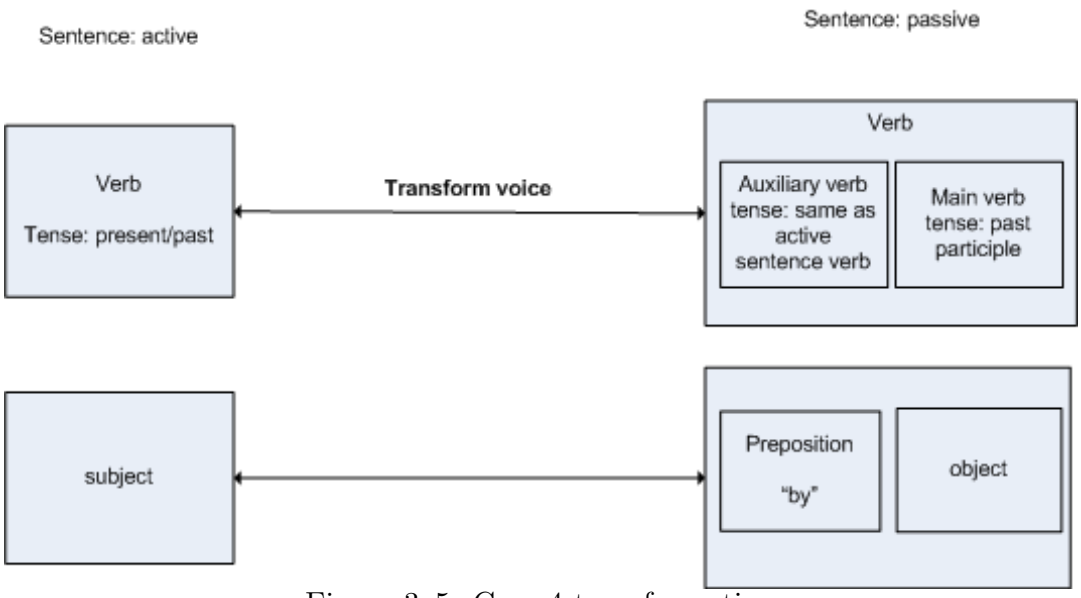

Figure 3–5: Case 4 transformations

Table 3–2 shows an example of an active and passive sentence pair, where the corresponding typed dependency transformations are demonstrated. The first row contains the example sentences, while the other rows contain the dependencies that show these transformations. Each row corresponds to each transformation. The auxpass dependency has no equivalent in the active sentence because it is generated only in the passive voice. It shows the relationship between the verb and its auxiliary.

Table 3–2: Table showing transformations for case 4

He rides the bike.	The bike is ridden by him.	Transformation
nsubj(rides, he)	agent(ridden, him)	Nominal subject dependency is transformed into an agent dependency, while the governors, which are the verbs, are transformed from active to passive, in this case, from simple present to past participle. In the same way, a personal pronoun is used as the dependent of the relation, and it is transformed from subjective to objective pronoun.
dobj(rides, bike)	nsubjpass(ridden, bike)	Direct object dependency is transformed to a passive nominal subject dependency, while the governors, which are the verbs, are transformed from active to passive, in this case, from simple present to past participle. The dependents remain the same.
	auxpass(ridden, is)	This dependency neither has a transformation, nor an equivalent one in the active sentence. This dependency is generated by the auxiliary verb existing in the passive voice. Since in the active voice there is not an auxiliary verb, this dependency is not generated in active sentences.

An equivalent pair of sentences in this case is a type of syntactic paraphrase [6].

3.3.5 Case 5 - Sentences in Simple Future

In this case, the semantic equivalence is defined between two sentences in simple future: one using "will" and the other using "is going to" or "are going to". For a pair of sentences to belong to this case, one of the sentences must have the auxiliary verb "will" and the other the "is going to" or "are going to" word sequence. "Will" in this case refers to the auxiliary verb only and not to any of the other meanings of this word.

Since the sentences have a different number of words, sentences with "is going to" or "are going to" are longer than sentences than those with "will", this case can be represented mathematically using specific parts of the sentences that transform or change.

The parser produces a set of collapsed typed dependencies which were used to analyze the sentence pair and determine equivalence between them. A nominal subject dependency (nsubj) and an auxiliary (aux) dependency are always present in a simple future sentence using "will", while a simple future sentence using "is going to" or "are going to" always has five collapsed typed dependencies: a nominal subject (nsubj), two auxiliary (aux) dependencies, an open clausal complement (xcomp), and a controlling subject (xsubj). In addition to the previously mentioned dependencies, the sentences may have other dependencies which must be identical in both sentences.

Figure 3–6 shows the transformations occurred when a sentence is transformed, in the simple future, from a "will" sentence to a "going to" sentence and viceversa.

Figure 3–6: Case 5 transformations

This case may be represented mathematically in the following form:

$$S_1 \iff S_2 \text{ iff } \forall_i(\alpha_i = \beta_i) \vee \exists_{i,j,k,k>j>i}(\alpha_i = \text{“}will\text{”}) \wedge (\alpha_i \in A_c) \wedge$$
$$(((\beta_i = \text{“}be\text{”}) \vee (\beta_i = \text{“}am\text{”}) \vee (\beta_i = \text{“}is\text{”}) \vee (\beta_i = \text{“}are\text{”})) \wedge$$
$$(\beta_j = \text{“}going\text{”}) \wedge (\beta_k = \text{“}to\text{”}))$$

Where:

$$S_1 = \sum_{i=1}^{n} \alpha_i$$
$$S_2 = \sum_{i=1}^{n} \beta_i$$

α_i represents the first sentence words and β_i represents the second sentence words. A_c is the auxiliary verbs set

3.3.6 Case 6 - Sentences in Future in the Past

In this case, the semantic equivalence is defined between two sentences in future in the past [44] [45]: one using "would" and the other using "was going to". For a pair of sentences belonging to this case, one of the sentences must have the auxiliary verb "would" and the other the word sequence "was going to" or "were going to". This case resembles Case 5. All that applies to Case 5 applies to this case too, except for the required words in the sentences.

As previous cases, this case may not be represented mathematically, in the same way as Cases 1, 2 and 3, because it depends on the semantics of the sentence. In addition, the pair of sentences has different quantity of words: "was going to" or "were going to" sentences are longer than those with "would". But this case can be represented mathematically using specific parts of the sentences that transform or change. The dependencies the parser produces for this case are very similar to those required for Case 5. In fact, the relationships are the same; the governors or dependents are what change.

Figure 3–7 shows the transformations occurred when a sentence is transformed, in the future in the past, from a "would" sentence to a "going to" sentence and viceversa.

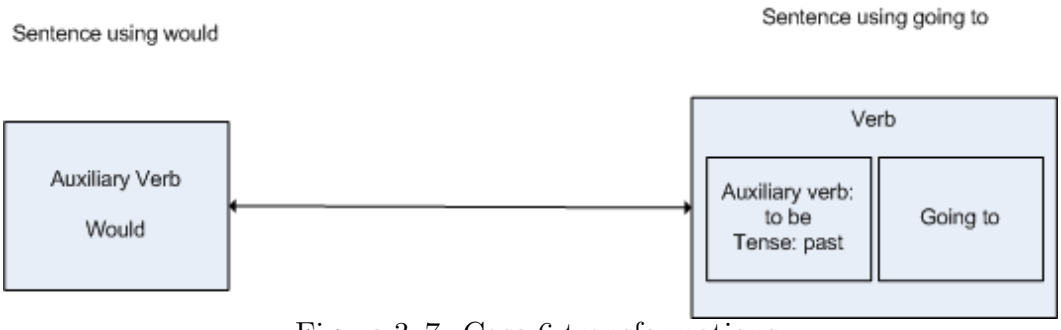

Figure 3–7: Case 6 transformations

This case may be represented mathematically in the following form:

$$S_1 \iff S_2 \text{ iff } \forall_i(\alpha_i = \beta_i) \vee \exists_{i,j,k,k>j>i}(\alpha_i = \text{“}would\text{”}) \wedge$$
$$(((\beta_i = \text{“}was\text{”}) \vee (\beta_i = \text{“}are\text{”})) \wedge$$
$$(\beta_j = \text{“}going\text{”}) \wedge (\beta_k = \text{“}to\text{”}))$$

Where:

$$S_1 = \sum_{i=1}^{n} \alpha_i$$
$$S_2 = \sum_{i=1}^{n} \beta_i$$

α_i represents the first sentence words and β_i represents the second sentence words.

3.3.7 Case 7 - Sentences using “can” and “be able to” verb forms

In this case, semantic equivalence is defined between two sentences, one using the verb “can” and the other using “be able to” verb form.

As in previous cases, this case may not be represented mathematically, in the same way as Cases 1, 2 and 3, because it depends on the semantics of the sentence. In addition, sentences in the pair have a different amount of words: sentences with “be able to” are longer than the ones using “can”. This case can be represented mathematically using specific parts of the sentences that transform or change.

The parser produces a set of collapsed typed dependencies which were used to analyze the sentence pair and determine equivalence between them. A nominal subject dependency (nsubj) and an auxiliary (aux) dependency are always present in sentences using "can", while sentences using "be able to" always have five collapsed typed dependencies: a nominal subject dependency (nsubj), a copula dependency (cop), an auxiliary (aux) dependency, and an open clausal complement (xcomp). These dependencies are required in the sentences within this case. Both kinds of sentences may have other dependencies, but those specified above must be present.

Figure 3–8 shows the transformations occurred when a sentence is transformed from a "can" sentence to a "be able to" sentence and viceversa.

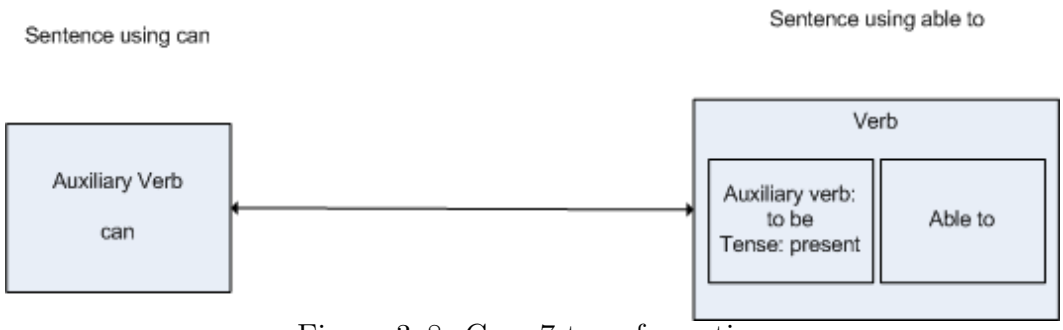

Figure 3–8: Case 7 transformations

This case may be represented mathematically in the following form:

$$S_1 \iff S_2 \text{ iff } \forall_i(\alpha_i = \beta_i) \vee \exists_{i,j,k,k>j>i}(\alpha_i = \text{“}can\text{”}) \wedge (\alpha_i \in A_c) \wedge$$
$$(((\beta_i = \text{“}be\text{”}) \vee (\beta_i = \text{“}am\text{”}) \vee (\beta_i = \text{“}is\text{”}) \vee (\beta_i = \text{“}are\text{”})) \wedge$$
$$(\beta_j = \text{“}able\text{”}) \wedge (\beta_k = \text{“}to\text{”}))$$

Where:

$$S_1 = \sum_{i=1}^{n} \alpha_i$$
$$S_2 = \sum_{i=1}^{n} \beta_i$$

α_i represents the first sentence words and β_i represents the second sentence words. A_c is the auxiliary verbs set

3.3.8 Negation with antonym

In this case, the semantic equivalence is defined between two sentences, where the second sentence contains a negation of an antonym of one or more words of the first sentence. Negated antonyms may appear one or more times in a sentence. Antonyms must be single-word antonyms, because, as Case 2, our system does not manage compound antonyms. This case, as Case 2, has the restriction that the only antonyms that it can recognize are those found in WordNet. If any word has an antonym that does not appear in WordNet, the system will not recognize the word as an antonym and the result will be incorrect. In the same way, if WordNet gives a compound antonym, the system will not manage it.

In this case, it is not always possible to determine semantic equivalence because antonyms are not always absolute. For example, "poor" is not the absolute antonym of "rich", *e.g.*, not poor is not the opposite of rich. A person can be neither rich nor poor, *i.e.* middle class. The same thing happens with pretty; being not pretty not necessary means being ugly. On the contrary, an example of an absolute antonym is "male". A person is either "male" or "female"; there are no intermediate levels.

This ambiguity occurs in other contexts. For example, in a verdict, if guilty cannot be proven beyond reasonable doubt, the statement is "not guilty" rather than "innocent".

As previous cases, this may not be represented mathematically, in the same way as Cases 1, 2 and 3, because it depends on the semantics of the sentence. In addition, the sentences in the pair have a different amount of words: the sentence with the negation is longer than the one with the antonym. This case can be represented mathematically using specific parts of the sentences that transform or change.

Figure 3–9 shows the transformations occurred when a sentence is transformed from a sentence using negation to a sentence using an antonym and viceversa.

Figure 3–9: Case 8 transformations

This case may be represented mathematically, as stated before, in the following form:

$$S_1 \iff S_2 \text{ iff } \forall_i((\alpha_i = \beta_i) \vee ((\alpha_i \in A_n \wedge \beta_i = \text{“}not\text{”} \wedge \beta_{i+1} \in A_n) \vee (\alpha_i \in D_T \wedge \alpha_{i+1} \in A_n \wedge \beta_i = \text{“}not\text{”} \wedge \beta_{i+1} = \alpha_i \wedge \beta_{i+2} \in A_n)))$$

Where:

$$S_1 = \sum_{i=1}^{n} \alpha_i$$
$$S_2 = \sum_{i=1}^{n} \beta_i$$

α_i represents the first sentence words and β_i represents the second sentence words.

A_n is the antonyms set

D_T is the determinants set

3.3.9 Combined cases

The following are formed by combinations of the previous cases. They can be expressed mathematically in the form of function composition. In this type of mathematical function, the output of one function becomes the input to another one. If a case is expressed as $f \circ g$, the function f represents the case which we refer to as master case, while the g function represents the case which we refer to as slave case. The master case has the most processing to determine the equivalence of a given sentence pair. Typically, it is one of the cases from 4 to 8. The master case determines if a given sentence pair belongs to the combined case, and determines if the sentence pair is equivalent. The slave case is in charge of processing only one part to determine the equivalence of a given sentence pair. Typically, it is either Case 2 or 3.

3.3.9.1 Case 9 - Combined case: Sentences in active and passive voice using synonyms

This case is a combination of Case 4, sentences in active and passive voice and Case 2, sentences using synonyms. In this case, the semantic equivalence is defined between two sentences, one in active voice and the other in passive voice, but using

one or more synonyms for words of the first sentence.

In this case, the function f represents Case 4 and the function g represents Case 2. All that applies to Case 4 can be applied to this case except that at least one synonym may be used in one of the sentences. The required dependencies of Case 4 are also required in this case. Synonyms can occur in the required dependencies or in any of the other dependencies of the sentence.

Table 3–3 shows an example of a pair of sentences of this case with the dependency transformations. As the table of Case 4, the first row contains the example sentences, while the other rows have the dependencies that show these transformations. Each row demonstrates each transformation. As shown, a synonym is used, i.e. “bike” for “bicycle”.

Table 3–3: Case Sentences in active and passive voice using synonyms transformation table

He rides the bicycle.	The bike is ridden by him.	Transformation
nsubj(rides, he)	agent(ridden, him)	Nominal subject dependency is transformed into an agent dependency, while the governors, which are the verbs, are transformed from active to passive, in this case, from simple present to past participle. In the same way, a personal pronoun is used as the dependent of the relation, and it is transformed from subjective to objective.
dobj(rides, bicycle)	nsubjpass(ridden, bike)	Direct object dependency is transformed to a passive nominal subject dependency, while the governors, which are the verbs, are transformed from active to passive, in this case, from simple present to past participle. Also, synonyms are used as the dependents of the relationship.
	auxpass(ridden, is)	This dependency does not have a transformation in the active sentence. This dependency is generated by the auxiliary verb existing in the passive voice.

3.3.9.2 Case 10 - Combined case: Sentences in Simple Future using synonyms

This case is a combination of Case 5, sentences in simple future and Case 2, sentences using synonyms. In this case, the semantic equivalence is defined between two sentences in simple future: one using "will" and the other using "is going to", but using one or more synonyms for words of the first sentence.

For this case, the function f represents Case 5 and the function g represents Case 2. All that applies to Case 5 can be applied to this, except that at least one synonym may be used in one of the sentences. Synonyms can occur in the required dependencies or in any of the other dependencies.

3.3.9.3 Case 11 - Combined case: Sentences in Future in the Past using synonyms

This case is a combination of Case 6, sentences in future in the past and Case 2, sentences using synonyms. In this case, the semantic equivalence is defined between two sentences in future in the past: one using "would" and the other using "was going to", but using one or more synonyms for words of the first sentence.

Function f represents Case 6 and the function g represents Case 2. All that applies to Case 6 can be applied to this case except that at least one synonym may be used in one of the sentences. The required dependencies of Case 6 are also required in this case. Synonyms can occur in any of the dependencies.

3.3.9.4 Case 12 - Combined case: Sentences using "can" and "be able to" verb forms using synonyms

This case is a combination of Case 7, sentences using "can" and "be able to" verb forms, and Case 2, sentences using synonyms.

The function f represents Case 7 and the function g represents Case 2. All that applies to Case 7 can be applied to this case except that at least one synonym may be used in one of the sentences. The required dependencies of Case 7 are also required in this case. Synonyms can occur in any of the dependencies.

3.3.9.5 Case 13 - Combined case: Negation with antonym using synonyms

This case is a combination of Case 8, negation with antonym and Case 2, identical sentences using synonyms. In this case, the semantic equivalence is defined between two sentences where one sentence contains a negation of one or more antonyms in the other sentence; in this case, however, one or more synonyms can be included in the sentence pair. As Case 8, antonyms must be single-words because the system does not manage compound antonyms.

The function f represents Case 8 and the function g represents Case 2. All that applies to Case 8 can be applied to this case except that at least one synonym has to be used in one of the sentences. The required dependencies of Case 8 are also required in this case. Synonyms cannot occur in the required negation dependency because it would not make sense. Synonyms can appear in the non required dependencies of this case or in the dependent of the required nominal subject.

3.3.9.6 Case 14 - Combined case: Simple Future using contractions

This case is a combination of Case 5, sentences in simple future and Case 3, identical sentences using contractions. In this case, the semantic equivalence is defined between two sentences in simple future: one using "will" and the other using "is going to", but using one or more contractions for auxiliary verbs of the first sentence.

The function f represents Case 5 and the function g represents Case 3. All that applies to Case 5 can be applied to this, except that at least one contraction may be used in one of the sentences. Contractions can occur in any of the non-required sentence dependencies.

In this chapter we have presented fourteen cases of semantic equivalence developed for this thesis. The following chapter will discuss the methodology of our work.

CHAPTER 4
METHODOLOGY

4.1 General Description

This chapter explains the methodology used in this work, focusing on the way each module was designed, developed, or used.

4.2 Tools

One of the main components of this work is the Stanford Natural Language Parser tool which was used to extract the grammatical information from the sentences. Another tool used in this work is WordNet, a lexical database used to find synonyms and antonyms. Java is the programming language used to code the experiments of this work. Java, the Stanford Natural Language Parser, WordNet, and other tools were integrated to develop the algorithms necessary for this thesis.

4.2.1 Stanford Natural Language Parser

The Stanford Natural Language Parser, hereinafter referred to as the parser, is a statistical, also known as probabilistic parser, which produces the grammatical information of a text in different formats. This parser was developed in the 1990's and is currently available for different languages including Chinese. For this work, we are using the English language parser. This parser is written in Java, open source and licensed under the GNU General Public License.

This parser uses the Penn Treebank parsed corpus which annotates phrase structures.

The parser extracts the grammatical structure from the text in different manners. One of them is extracting the Part-Of-Speech word tags. In this approach all words are tagged using the part-of-speech tag set from Penn Treebank. An example of this approach using the sentence "My dog also likes eating sausage." is shown below.
My/PRP$ dog/NN also/RB likes/VBZ eating/VBG sausage/NN ./.

Another approach in which the parser extracts the grammatical structure of the text is generating a tree structure of the sentences. An example of this approach using the same example sentence is shown below.

```
(ROOT
    (S
       (NP (PRP$ My) (NN dog))
       (ADVP (RB also))
       (VP (VBZ likes)
            (S
               (VP(VBG eating)
                    (NP (NN sausage)))))
       (. .)))
```

The Stanford Natural Language Parser has implemented 55 grammatical relationships, also known as typed dependencies. These grammatical relationships are presented as triples composed of the name of the relation, and the governor and

the dependent as parameters. Typed dependencies are another approach to extract the grammatical structure of texts. The parser generates the sentence typed dependencies in five categories: basic, collapsed, collapsed with propagation of conjunct dependencies, collapsed preserving a tree structure, and non-Collapsed dependencies. These types of dependencies were discussed in the Theoretical Framework. The dependencies used in this work were the collapsed dependencies. An example of collapsed dependencies representation using the sentence "My dog also likes eating sausage." is shown below.

poss(dog-2, My-1)

nsubj(likes-4, dog-2)

advmod(likes-4, also-3)

xcomp(likes-4, eating-5)

dobj(eating-5, sausage-6)

The parser version we are using for this work is 1.6.3.

4.2.2 WordNet

WordNet [46] is a lexical database created at Princeton University [47]. WordNet is composed of synsets, which are synonym sets, also known as collocations, which are logical groups where the information is organized. Synsets are divided in four categories: nouns, verbs, adjectives, and adverbs. From these, the nouns category is the more extensive. The version 3.0 of this database, which has a total of 117659 synsets, were used.

In this work, WordNet was used to get the synonyms and/or antonyms for Case 2, Identical Sentences Using Synonyms, Case 8, Negation with antonym, and

for combined cases where at least one of the cases used synonyms or antonyms.

JAWS, the Java API for WordNet was used. This API is a library compatible with WordNet 2.0 and 3.0 and with Java 1.4 or later versions.

4.2.3 Java Peculiarities

The algorithms were developed using Java version 1.6. This language was selected for compatibility and easy integration with the NLP parser and the libraries to connect to Wordnet. In addition, Java is cross platform, and it should work seamlessly in any operating system which has the Java Virtual Machine (JVM) installed.

One of the peculiarities of the Java language is that it does not allow multiple inheritance, as other programming languages such as C++ do. To simulate multiple inheritance in Java, the language provides the combination of inheritance and interfaces. Multiple inheritance became an issue in the development of the algorithms for the combined cases and code had to be duplicated. In order to design and develop the combined cases, we have to take one case as the "master class" and the other case as the "slave class". The "slave class" was selected as the class that had fewer methods to inherit. In the same way, the "master class" was selected as the class that had more methods to inherit. In this way, slave cases inherit from the master class and implement the combined class code from one of the classes and implemented code from a Java interface for the other class.

4.2.4 SuperCSV1.52

SuperCSV is a library tool that reads and writes content of a CSV file. The version used is SuperCSV 1.52. We integrated this library in the Java Project to read and write the CSV files created to simulate and manage the verbs, pronouns

and contractions equivalence databases.

4.3 Design Approach

Semantic equivalence was determined designing and developing a series of algorithms based on cases. These cases were defined selecting different common forms to represent sentences. In this work, fourteen cases were designed and developed. In all the cases the typed dependencies collapsed were obtained from the parser.

Figure 4–1 shows a graphical representation of how the system analyzes a given pair of sentences to determine equivalence.

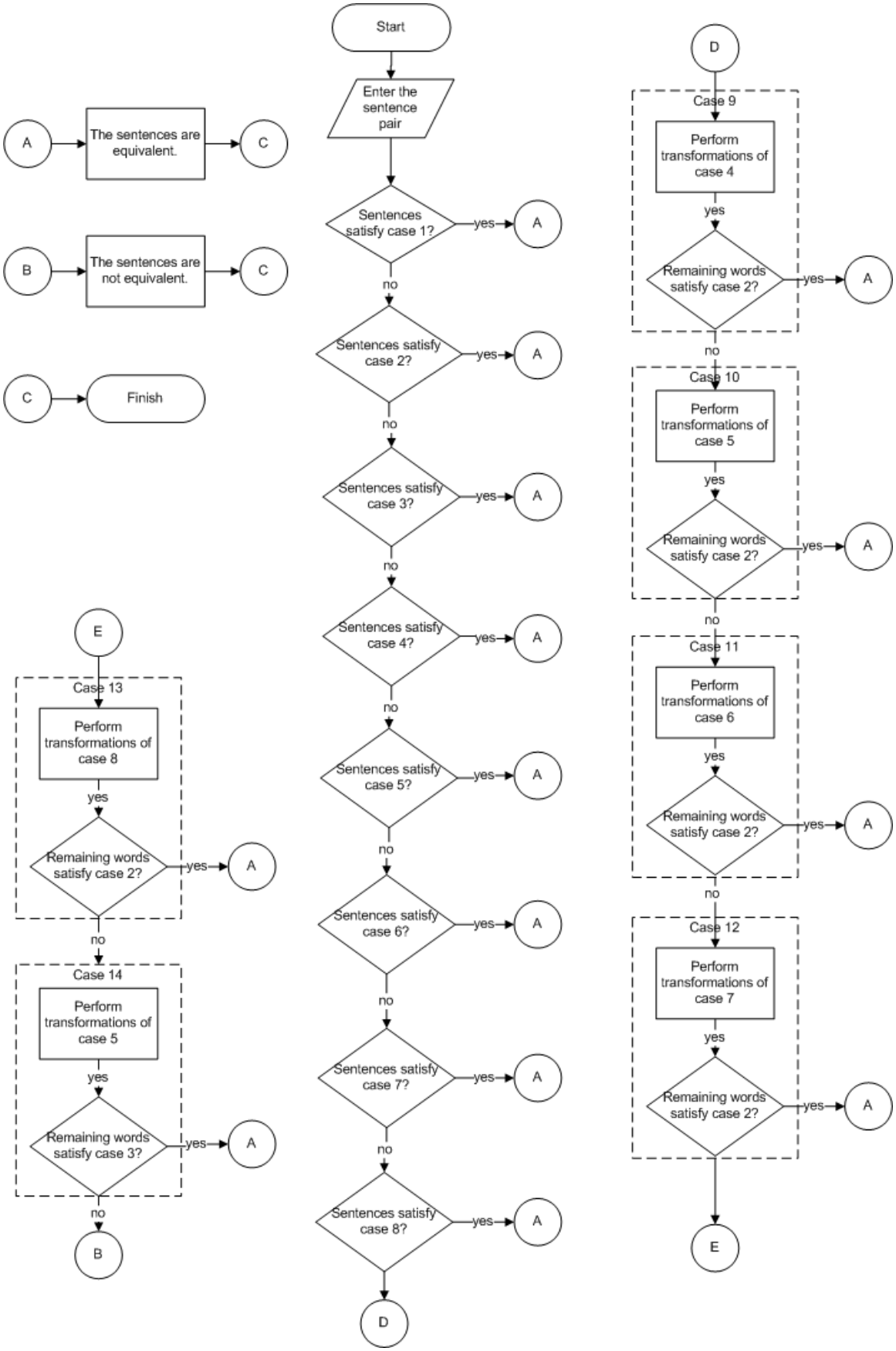

Figure 4–1: Graphical representation of the process

In all cases, the sentence pair is analyzed looking at the typed dependencies collapsed of both sentences. In many cases the sentences have some dependencies which we named template dependencies. These dependencies are those which always appear in such sentences, following a specific pattern. Each case has its own rules to determine which of the type dependency collapsed become template dependencies.

In the cases where template dependencies are defined, to determine equivalence between the sentence pair we compared the template dependencies of both sentences and the remaining dependencies that the sentences had. If the template dependencies complied with the restrictions of the case, the template dependencies were equivalent. For the non-template dependencies, they must be identical for the sentences to be equivalent. For the combined cases using synonyms, the template dependencies and the remaining ones, must be equivalent. By equivalent we mean that the dependencies must be either identical or contain synonyms.

In the following subsections all the cases will be explained in detail.

4.3.1 Case 1 - Identical Sentences

In this case the two sentences of a pair must be identical. For this case, two approaches were considered. The first one compares the two sentences as strings, and if the strings are identical, the sentences are equivalent. Once one word is found not identical, all the remaining words are not analyzed, and the sentence pair is considered not equivalent. This approach is a simple algorithm, easy to implement, and fast because there is no need to call the parser. The second one compares the typed dependencies collapsed of both sentences in terms of relation name, governor, and dependent. If the dependencies are identical, the sentences are declared equivalent. Once a dependency is found not equivalent, all the remaining dependencies

are not analyzed and the sentence pair is not considered equivalent under this case. Although at first sight, the first approach may be considered a better choice, the second approach were selected because with the first one, one simple type error, such as an extra space between words in one of the sentences, would make the system fail and give a false negative.

An example of an equivalent sentence pair for this case is {"I am a computer engineering student.", "I am a computer engineering student."}.

4.3.2 Case 2 - Identical Sentences using synonyms

In this case the two sentences of the pair must be identical but may use synonyms of one or more words. A restriction of this case is that the only synonyms recognized are those from WordNet, which gives a set of synonyms of a given word in a Java array. Any other synonym is not recognized and the system fails. Another restriction is that the system only manages single word synonyms, although WordNet gives compound synonyms.

For this case, the same two approaches described for the Case 1 were considered. As in the Case 1, the second approach was chosen. The approach compares the typed dependencies collapsed of both sentences in terms of relation name, governor, and dependent. While the relationship names must be identical, the governors and dependents may not necessarily be. If in a relationship pair, the governors are different, WordNet is called to give the synonyms of the governor of the second relation. The same occures with the dependents. If in a relationship pair, the dependents are different, WordNet is called to give the synonyms of the dependents of the second relation. If the governor of the first relation is identical to the governor of the second relation, or if the governor of the first relation is a synonym of the governor of the

second relation, and if the dependent of the first relation is identical to the dependent of the second relation, or if the dependent of the first relation is a synonym of the dependent of the second relation, the relationships are equivalent. If at the end all the relationships were equivalent, the sentence pair is equivalent; otherwise, the sentence pair is not equivalent under this case.

An example of an equivalent sentence pair for this case is {"My car is blue.", "My automobile is blue."}.

4.3.3 Case 3 - Identical Sentences using contractions

In this case the two sentences of the pair may use contractions of auxiliary verbs. A restriction of this case is that the only contractions recognized ae those stored in a CSV file denoted the contractions file. The contractions file contains the contractions and the corresponding auxiliary verbs in two columns. One column stores the contractions and the other stores the equivalent auxiliary verbs.

For this case, the same two approaches for the first two cases were considered. The second approach compares the typed dependencies collapsed of both sentences in terms of relation name, governor, and dependent. While the relationship names must be identical, the governors and dependents may not necessarily be. If in a relationship pair, the governors are different, the CSV file is read to find the corresponding contraction of the word of the second sentence or the word corresponding to the contraction found. The same occurs with the dependents. If in a relation pair, the dependents are different, the CSV file is read to find the corresponding contraction of the word of the second sentence or the word corresponding to the contraction found. If the governor of the first relationship is identical to the governor of the second relationship or if the governor of the first relationship is a contraction of the

governor of the second relationship, and if the dependent of the first relationship is identical to the dependent of the second relationship or if the dependent of the first relationship is a contraction of the dependent of the second relation, the relationships are equivalent. If at the end all the relationships are equivalent, the sentence pair is equivalent.

An example of an equivalent sentence pair for this case is {"I am a computer engineer.", "I'm a computer engineer."}.

4.3.4 Case 4 - Sentences in active and passive voice (Active vs. Passive sentences)

In this case, the sentences are analyzed by looking at the typed dependencies collapsed of both sentences and realizing that active and passive sentences have some dependencies which we named template dependencies. These dependencies are those which always appear in such sentences, following a pattern. In active sentences, the combination of nominal subject dependency and direct object dependency always appear. These are the template dependencies for the active sentence. These dependencies are not the only condition to become template dependencies. To become template dependencies, the nominal subject dependency, "*nsubj*", and the direct object dependency, "*dobj*", share the same governor. This governor must be the same in the template dependencies of the passive sentence but transformed. The governor of these dependencies in the passive sentence is a verb which appears in passive voice. In the passive sentences, the combination of passive nominal subject dependency, "*nsubjpass*", passive auxiliary dependency, "*auxpass*", and agent dependency, "*agent*", always appear and share the same governor. We assumed that the sentences are simple, not compound sentences. With this assumption, the template dependencies for the passive sentences appear only once, and once they

appear, they are set as passive template dependencies.

To determine equivalence in the sentence pair, the template dependencies of both sentences as well as the remaining dependencies of the sentences are compared. For the template dependencies, the governor of the active sentence and the passive sentence dependencies is the same, but transformed from the active to the passive voice, as previously stated. The "*nsubjpass*" dependent is the same as the "*dobj*" dependent, and the "*agent*" dependent is the same as the "*nsubj*" dependent. Also, the "*agent*" and the "*nsubj*" dependents are identical if they are nouns. If they are pronouns, they are transformed from subjective to objective. In addition, the "*auxpass*" dependent is a conjugated form of the verb "to be" in the same tense as the main verb of the active sentence. This means that the "*auxpass*" dependent and the "*nsubj*" governor must have the same verb tense. If all these conditions are satisfied, the template dependencies are equivalent. The non-template dependencies must be identical for the sentences to be equivalent.

Table 4–1 shows an example of the transformations in this case.

Table 4–1: Example of Case 4 Transformations

Sentence Pair	He rides the bicycle.	The bicycle is ridden by him.
Template typed dependencies collapsed	nsubj(rides-2, He-1) det(bicycle-4, the-3) dobj(rides-2, bicycle-4)	det(bicycle-2, The-1) nsubjpass(ridden-4, bicycle-2) auxpass(ridden-4, is-3) agent(ridden-4, him-6)

This example shows how from the typed dependencies collapsed the template and the non-template dependencies can be extracted. The template dependencies for the active sentence are "*nsubj(rides-2, He-1)*" and "*dobj(rides-2, bicycle-4)*", while the non-template dependency is "*det(bicycle-4, the-3)*". The template dependencies

for the passive sentence are "*nsubjpass(ridden-4, bicycle-2)*", "*auxpass(ridden-4, is-3)*", and "*agent(ridden-4, him-6)*" while the non-template dependency is "*det(bicycle-2, The-1)*". We can note that the governors of "*nsubj*" and "*dobj*" are the same, while the transformation of the verb in sentences is seen in the governors of these dependencies, and the governors of the passive dependencies. In addition, we can note that the "*nsubj*" and "*agent*" dependents are equivalent pronouns, and so the relations are equivalent. Also we can note that "*dobj*" and "*nsubjpass*" dependents are the same, so the relations are equivalent. Another feature to note is that the tense of the active and passive sentences is the same. This can be observed from the tense of the verb in the active sentence, which is the governor of either the "*nsubj*" or "*dobj*" dependencies, and the tense of the auxiliary verb in the passive sentence, which is the dependent of the "*auxpass*" dependency.

4.3.5 Case 5 - Sentences in Simple Future

In this case the two sentences must be a simple future pair, which means that one sentence must be in the simple future tense using the auxiliary verb "will" and the other using the future verb phrase "going to". The phrase "going to" is preceded by the auxiliary verb "to be" conjugated in present tense. The verb "to be" is part of the "going to" future verb phrase. To determine if the sentence pair has the characteristic mentioned above, the sentence that contains the auxiliary verb "will" is called "will type", and the other sentence of the pair is called "going to type".

In this case, the sentences are analyzed looking at the typed dependencies collapsed of both sentences. In "will type" sentences, the combination of nominal subject dependency, "*nsubj*", and auxiliary dependency, "*aux*", always appear. These are the template dependencies for the "will type" sentence. In the "going to type" sentences, the combination of a nominal subject dependency, "*nsubj*", two auxiliary

dependencies, "*aux*", an open clausal complement dependency, "*xcomp*", and a controlling subject dependency, "*xsubj*", always appear.

To become template dependencies the "will type" aux dependency must have the "will" auxiliary verb as its dependent, while in the "going to" sentence, the "*nsubj*" dependency, the "*xsubj*" dependency, and one of the "*aux*" dependencies must have the "going" verb as its governor. The "going to" "*aux*" dependency referred in the previous sentence must have the "to" preposition as the dependent. We called this dependency the "second" "going to" "*aux*" dependency. If it is called the "second" we can infer that a "first" "*aux*" should exist. The "first" "going to" "*aux*" template dependency is the one that has the "going" verb as the governor and the conjugated "to be" auxiliary verb as the dependent. This auxiliary verb provides information about the person/number of the subject in the sentence. The following "going to" template dependencies must have the "going" verb as the governor: the "*nsubj*", the "first" "*aux*", and the "*xcomp*". Other restrictions to become template dependencies were that the "will type" "*nsubj*" dependency and the "will type" "*aux*" dependency must share the same governor. This governor must be the same that the governor of both, the second "going to" "*aux*" dependency and the "*xsubj*" dependency, and also must be the same as the dependent of the "*xcomp*" dependency. Another restriction is that the dependent of the "will type" "*nsubj*" dependency must be the same as the dependents of both, the "going to" type "*nsubj*" and "*xsubj*" dependencies.

These template dependencies can be determined using the form shown in Table 4–2:

Table 4–2: Case 5 template dependencies

"will" type sentence	"going to" type sentence
nsubj(X, S) aux(X, will)	nsubj(going, S) xsubj(X, S) aux(going, to_be_verb_form) aux(X, to) xcomp(going, X)

Table 4–3 shows an example of an equivalent simple future sentence pair with their respective template dependencies.

Table 4–3: Template dependencies of an equivalent example sentence pair

Sentence Pair	Sue will make John's birthday cake.	Sue is going to make John's birthday cake.
Template typed dependencies collapsed	nsubj(make-3, Sue-1) aux(make-3, will-2)	nsubj(going-3, Sue-1) xsubj(make-5, Sue-1) aux(going-3, is-2) aux(make-5, to-4) xcomp(going-3, make-5)

4.3.6 Case 6 - Sentences in Future in the Past

This case is very similar to the "Sentence in simple future" case. The main differences between this case and the other are the use of the auxiliary verb "would" instead of "will" in one of the sentences. Also, the "to be" auxiliary verb that precedes the "going to" future verb phrase is substituted by its past tense, i.e., it is conjugated as "is" or "are", instead of "am", "was" or "were".

In this case, the sentences are analyzed in the same way as was done in Case 5. The difference is the dependency that uses the auxiliary verb "will" in Case 5, use the auxiliary verb "would" in this case, and the tense of the auxiliary verb on

the “going to” sentence is past. The template dependencies generated are the same, except for the mentioned change in words.

The template dependencies of this case are determined using the form shown in Table 4–4.

Table 4–4: Case 6 template dependencies

“would” type sentence	“going to” type sentence
nsubj(X, S) aux(X, would)	nsubj(going, S) xsubj(X, S) aux(going, to_be_verb_form) aux(X, to) xcomp(going, X)

Table 4–5 shows an example of an equivalent future in the past sentence pair with their respective template dependencies.

Table 4–5: Template dependencies of an equivalent example sentence pair

Sentence Pair	Sue would make John’s birthday cake.	Sue was going to make John’s birthday cake.
Template typed dependencies collapsed	nsubj(make-3, Sue-1) aux(make-3, would-2)	nsubj(going-3, Sue-1) xsubj(make-5, Sue-1) aux(going-3, was-2) aux(make-5, to-4) xcomp(going-3, make-5)

4.3.7 Case 7 - Sentences using “can” and “be able to” verb forms

In this case, one sentence must have the ability modal verb “can” and the other sentence must have the ability modal verb “able to”. The “able to” is preceded by the auxiliary verb “to be” conjugated. The sentence using the “can” modal auxiliary verb are called “can type” sentence, while the other sentence are called “able to type”.

In "can type" sentences, the combination of nominal subject dependency "*nsubj*", and auxiliary dependency "*aux*" always appear. These are the template dependencies for the "can type" sentence. These dependencies are not the only condition to become template dependencies. In "able to type" sentences, the combination of a nominal subject dependency "*nsubj*" , a copula dependency "*cop*", an auxiliary dependency, "*aux*", and an open clausal complement dependency, "*xcomp*" always appear.

To become template dependencies the "can type" "*aux*" dependency must have the auxiliary verb "can" as its dependent, while in the "able to" sentence, the "*nsubj*" dependency, the "*cop*" dependency, and the "*aux*" dependency must have the "*able*" modal verb as its governor. In the dependencies that we refer to in the previous sentence, the "*cop*" dependency must have the auxiliary verb "to be" conjugated as the dependent, while the "*aux*" dependency must have the "to" preposition as the dependent. Other restrictions to become template dependencies are that the "can type" "*nsubj*" dependency and the "can type" "*aux*" dependency must share the same governor. This governor must be the same as the governor of the "able to" "*aux*" dependency, and also must be the same as the dependent of the "*xcomp*" dependency. Another restriction is that the "can type" "*nsubj*" dependency and the "able to" "*nsubj*" dependency must have the same dependent.

These template dependencies can be determined using the form shown in Table 4–6.

Table 4–7 shows an example of an equivalent ability sentence pair with their respective template dependencies.

Table 4–6: Case 7 template dependencies

"can" type sentence	"able to" type sentence
nsubj(X, S) aux(X, can)	nsubj(able, S) cop(able, to_be_verb_form) aux(X, to) xcomp(able, X)

Table 4–7: Template dependencies of an equivalent example sentence pair

Sentence Pair	I can play piano.	I am able to play piano.
Template typed dependencies collapsed	nsubj(play-3, I-1) aux(play-3, can -2)	nsubj(able-3, I-1) cop(able-3, am-2) aux(play-5, to-4) xcomp(able-3, play-5)

4.3.8 Case 8 - Negation of antonym

In this case one sentence must have a negated antonym for one or more words of the other sentence. An example of an equivalent sentence pair for this case is {"The baby is a girl.", "The baby is not a boy"}.

In "antonym type" sentences, since the sentences can be of any type, there is not a particular dependency that appears in this sentence type, but looking at the template dependencies of the "negation type" sentence, the nominal subject dependency, "*nsubj*", is needed when comparing the "antonym type" sentence with the "negation". Therefore, the "*nsubj*" dependency becomes the template dependency for the "antonym type" sentence. In addition, in the "negation type" sentences, the combination of a nominal subject dependency, "*nsubj*", with a negation dependency, "*neg*", always appear. The "*neg*" dependency is always a template dependency. The condition is for the "*nsubj*" dependencies of both sentences.

To become template dependencies, the "negation type" "*nsubj*" dependency must have the same governor as the "*neg*" dependency, which has a negation word as its dependent, while the "antonym" "*nsubj*" dependency must have as its governor the antonym of the governor of the "negation type" template dependencies. In

addition, the "antonym" "*nsubj*" dependency and the "negation" "*nsubj*" dependency must share the same dependent.

These template dependencies can be determined using the following form:

Table 4–8: Case 8 template dependencies

"antonym" type sentence	"negation" type sentence
nsubj($\neg X$ [1] , Y)	nsubj(X, Y) neg(X, S)

Table 4–9 shows an example of antonym sentence pair with their respective template dependencies.

Table 4–9: Template dependencies of an equivalent example sentence pair

Sentence Pair	The baby is a girl.	The baby is not a boy.
Template typed dependencies collapsed	nsubj(girl-5, baby-2)	nsubj(boy-6, baby-2) neg(boy-6, not-4)

This case has the same restriction as the "Identical Sentences using synonyms". The only antonyms the system can recognize are those which come from WordNet. If any word has an antonym that does not appear in WordNet, and this word is used as an antonym in a sentence, the system will not recognize the word as an antonym and the result will be incorrect. In the same way, if WordNet gives a compound antonym, the system will not manage it. Another restriction is that the antonyms must be absolute, as explained in section 3.3.8.

4.3.9 Combined cases

The combined cases were formed from combinations of two of the previous cases. They were implemented simulating multiple inheritance, since this is not provided by the Java language. Multiple inheritance was simulated using single inheritance and Java interfaces. The two cases involved in any combination We will referred to

as master and slave. The case which had the most processing in the combination were selected as the master, and the case with the least processing as the slave. The master case was the superclass of the combined case, while the slave was the interface to be implemented. As the slave case becomes a Java interface, the code was duplicated in each case where needed.

In the combined cases using synonyms, at least one synonym must be used, but synonyms do not include verbs, while, in the combined cases using contractions, at least one contraction must be used. The cases inherited the behavior of the master case and implemented the algorithm that worked with the synonyms or contractions in the slave case, depending on the case. Since the cases inherited the behavior of the master case, it was first determined if the sentence pair belonged to the master case; if true, the equivalence of the sentence pair was then determined in the slave case.

The dependencies of the sentences, both template and non-template, may be affected by the use of synonyms or contractions. The synonyms algorithm determines if any pair of words are synonyms, and if so, the system accepts a pair of dependencies as equivalent, even if they are not identical, or do not have identical governors or dependents. The contractions algorithm works in a similar way, but using contractions instead of synonyms. The template dependencies must be affected only when requiring identical governors or identical dependents between both sentences. In this case, the governors or the dependents must be synonyms or contractions, depending on the case, for the sentences to be equivalent. This only happens, when comparing dependencies between sentences.

4.3.9.1 Case 9 - Combined case: Sentences in active and passive voice (Active vs. Passive sentences) using synonyms

In this case, the two sentences must be an active-passive pair and the master case is the "Sentences in active and passive voice".

Table 4–10 shows an example of an equivalent active-passive sentence pair using synonyms with their respective template dependencies.

Table 4–10: Template dependencies of an equivalent example sentence pair

Sentence Pair	He rides the bicycle.	The bike is ridden by him.
Template typed dependencies collapsed	nsubj(rides-2, He-1) dobj(rides-2, bicycle-4)	nsubjpass(ridden-4, bike-2) auxpass(ridden-4, is-3) agent(ridden-4, him-6)

4.3.9.2 Case 10 - Combined case: Sentences in Simple Future using synonyms

In this case, the "Sentences in Simple Future" case is the master case, therefore the two sentences of the sentence pair must be a simple future pair. The synonyms include neither the auxiliary verb "will" nor "going to".

Table 4–11 shows an example of an equivalent simple future sentence pair using synonyms with their respective template dependencies.

Table 4–11: Template dependencies of an equivalent example sentence pair

Sentence Pair	This movie will win several Academy Awards.	This film is going to win several Academy Awards.
Template typed dependencies collapsed	nsubj(win-4, movie-2) aux(win-4, will-3)	nsubj(going-4, film-2) xsubj(win-6, film-2) aux(going-4, is-3) aux(win-6, to-5) xcomp(going-4, win-6)

4.3.9.3 Case 11 - Combined case: Sentences in Future in the Past using synonyms

In this case, the "Sentences in Future in the Past" case is the master case, and therefore the two sentences must be a future in the past pair. Similarly to Case 10 , the synonyms include neither the auxiliary verb "would" nor "going to".

Table 4–12 shows an example of an equivalent future in the past sentence pair with their respective template dependencies.

Table 4–12: Template dependencies of an equivalent example sentence pair

Sentence Pair	This movie would win several Academy Awards.	This film was going to win several Academy Awards.
Template typed dependencies collapsed	nsubj(win-4, movie-2) aux(win-4, would-3)	nsubj(going-4, film-2) xsubj(win-6, film-2) aux(going-4, was-3) aux(win-6, to-5) xcomp(going-4, win-6)

4.3.9.4 Case 12 - Combined case: Sentences using "can" and "be able to" verb forms using synonyms

In this case, the "Sentences using "can" and "be able to" verb forms" case is the master case, therefore the two sentences must be an ability sentence pair where one sentence must be a "can" sentence and the other a "be able to" sentence. Similarly to Case 10 and Case 11 , the synonyms do not include either "can" or "able to".

Table 4–13 shows an example of an equivalent ability sentence pair with their respective template dependencies.

Table 4–13: Template dependencies of an equivalent example sentence pair

Sentence Pair	I can drive Susan's car when she is out of town.	I am able to drive Susan's automobile when she is out of town.
Template typed dependencies collapsed	nsubj(drive-3, I-1) aux(drive-3, can-2)	nsubj(able-3, I-1) cop(able-3, am-2) aux(drive-5, to-4) xcomp(able-3, drive-5)

4.3.9.5 Case 13 - Combined case: Negation of antonym using synonyms

In this case, the "Negation of antonym" case is the master case; therefore the sentences must be a negation of antonym sentence pair, but using at least one synonym.

Table 4–14 shows an example of an equivalent Negation of antonym case with their respective template dependencies.

Table 4–14: Template dependencies of an equivalent example sentence pair

Sentence Pair	The baby is a girl.	The infant is not a boy.
Template typed dependencies collapsed	nsubj(girl-5, baby-2)	nsubj(boy-6, infant-2) neg(boy-6, not-4)

4.3.9.6 Case 14 - Combined case: Simple Future using contractions

In this case, the "Sentences in Simple Future" case is the master case; therefore the two sentences of the sentence pair must be a simple future pair, but using at least one contraction. At the time, the contractions include neither the auxiliary verb "will" nor "going to".

Table 4–15 shows an example of an equivalent simple future sentence pair using contractions with their respective template dependencies.

Table 4–15: Template dependencies of an equivalent example sentence pair

Sentence Pair	I do not know if I will travel next month.	I don't know if I am going to travel next month.
Template typed dependencies collapsed	nsubj(travel-8, I-6) aux(travel-8, will-7)	nsubj(going-8, I-6) xsubj(travel-10, I-6) aux(going-8, am-7) aux(travel-10, to-9) xcomp(going-8, travel-10)

Table 4–15 shows that the sentence pair belongs to Case 5 (Simple Future), but the contraction is not shown there because it belongs to the non-template typed dependencies. The dependencies where the contraction equivalence is shown are:

neg(know-4, not-3)

neg(know-4, n't-3)

There, the system determines the equivalence between "*not*" and "*n't*". Once it is determined that the sentence pair belonges correctly to Case 5 and there exist a contraction that is recognized and determined as correct (Case 3), the system recognizes the sentence pair as equivalent in this case (Case 14).

4.4 Database Simulation

A database was simulated using text files in comma separated values (CSV) form. Three CSV files were used for:

1. Verb equivalence;
2. Pronoun equivalence; and
3. Contraction equivalence.

When pronouns, verbs or contractions are used, the system reads the data from the respective file or files. The use of each of these files is explained below. Additional equivalence tuples can be easily added to these files if needed. The SuperCSV API was the tool in charge of simulating the operation and management of the database.

With the verbs file we can determine if two verbs are equivalent in terms of tense, i.e., if both verbs are the same but conjugated differently. For example, if we have an equivalent pair of sentences in active and passive voice that use the verb "to drive" conjugated as "drives" for the active voice and "driven" for the passive voice, the system will detect that both verbs are equivalent in the active and passive sentence pair using the respective file. If an equivalent sentence pair was tested and the sentences use a verb not stored in the file, the system will fail the equivalence test. Something similar could occur with pronouns and contractions.

In the case of pronouns, only personal pronouns were considered. These were used for the case of active versus passive sentences. Personal pronouns in archaic forms were not considered. The pronouns file was used to determine if two pronouns were in their objective and subjective form. They must agree in person, number and gender.

Something similar occurs with the contractions. The contractions file contains equivalent contraction pairs. Examples of contraction pairs are "am" and "'m", "are" and "'re", and "is" and "'s".

4.5 Tests

Preliminary tests were conducted using a few sentences created by us, to test each case separately, letting us find the system bugs, and know how the system could be improved. These sentences were stored in a text file or entered by console. The text file referred to consisted of 87 test sentences, while by console any sentence pair would be tested.

For the final system tests, part of the Microsoft research paraphrase corpus, MSRP, was used as described in the Tests and Results chapter.

CHAPTER 5
TESTS AND RESULTS

This chapter presents the tests performed to all the cases and their results. A group of sentence pairs, to test each case, was developed. The results and a discussion of each one are presented.

5.1 Tests

For the development of the system tests, we used the MSRP corpus, which has a total of 5,801 sentence pairs. Each sentence pair, as its name suggests, consists of 2 sentences. This corpus was developed for entailment purposes, not for equivalence, reason why we had to modify it in order to use of it as an equivalence corpus. As this thesis works on equivalence cases, a set of pairs of sentences were selected from the corpus, to create a new one, a corpus of equivalent sentences. This new corpus has 451 pairs of sentences, some of which were modified to work on more than one case. Therefore, some sentences are repeated but modified differently in order to belong to different cases. In this new corpus, 395 pairs of sentences are equivalent pairs according to the studied cases, while 56 are non-equivalent.

5.2 Results and Discussion

For the each studied equivalence case, a certain quantity of pairs of sentences in this corpus is equivalent, while other pairs of sentences are non-equivalent. Of the pairs of sentences that were non-equivalent, some can be false negatives, while some others were really non-equivalent. These results allow measuring the percent

of successful identification of equivalent or non-equivalent pairs.

For each case the results were displayed in tables that have three columns. The first column shows the total number of sentence pairs for the case. The second and third columns were divided in two sub-columns. The second column shows the pairs that were found equivalent, either because the pair was truly equivalent, left sub-column, or the system gave a false positive, right sub-column. The third column shows the pairs that were classified as non-equivalent, where the left sub-column displays the pairs that were truly non-equivalent, and the right sub-column presents the pairs that resulted as false negatives.

5.2.1 Base cases results and discussion

In this section, the results of the base cases are shown. Base cases refer to those which are not combined with any other.

Table 5–1 shows the results of the sentences which belong to Case 1.

Table 5–1: Case 1 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
32	22	0	11	0

In this case there were neither false positives, nor false negatives. Of the non-equivalent sentence pairs tested, 5 had minimal changes between the sentences of the pair, 3 pairs had entailed sentences directly from the MSRP corpus, and 3 pairs had non-related sentences directly from the MSRP corpus. From the 3 pairs of the non-related sentences, 1 pair had sentences related to the same topic (cancer) but otherwise non-equivalent.

Table 5–2: Case 2 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
60	48	0	6	6

Table 5–2 shows the results for Case 2. Of all the sentence pairs in this case, 49 had one synonym while 11 had more than one synonym in the sentence pair.

In this case, not all synonyms were available on WordNet which resulted in false negatives. There were no false positives. The WordNet synonyms that our system recognizes are nouns. As our system cannot recognize adjective nor verb synonyms, false negatives were caused when a synonym was used in one of these parts of speech. Three of the six false negatives found in this test were due to this restriction.

Another failure obtained was because our system does not recognize numeric equivalence. For example, our system cannot recognize numeric equivalence such as "1000 millions" and "1 billion", "100 milligrams" and "0.1 gram", or others equivalences such as these. One of the six false negatives found in this test was for this restriction.

In the sentences that have more than one synonym, the system recognized all synonyms and performed well. If one word did not have a synonym in the other sentence, the sentence pair was automatically classified as non-equivalent in Case 2, although the other words had synonyms. Two of the six false negatives found in this test were caused by this.

Table 5–3: Case 3 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
98	97	0	10	1

Table 5–3 shows the results for Case 3. Of all sentence pairs in this case, 79 pairs had one contraction and 11 had more than one contraction.

In this case, the system produced one false negative as a response to a pair of sentences where the NLP parser failed to recognize " 's" as the contraction of "has", the auxiliary verb , and instead it confused it with a possessive. The error of the NLP parser led the system to find the sentence pair as non-equivalent.

Table 5–4 presents the results for Case 4.

Table 5–4: Case 4 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
32	15	0	2	15

One problem which caused false negatives in this case was that in some long sentences, the parser did not always set the typed dependencies correctly, and this issue affected the results of the system. If the template dependencies were not set correctly by the parser, the system was not able to find equivalence between these dependencies. One such case occurred when the parser used the generic dependency "dep". Other false negatives occurred when the governors or dependents produced by the parser were not correct, even if the dependency names were correct. Something similar happened if any non-template dependency pair was not equivalent. The sentence pair was automatically set as non-equivalent pair. Some experimentation helped to discover some errors in some parser results. For example, the sentence pair *"Five more human cases of West Nile virus, were reported by the Mesa County Health Department on Wednesday. The Mesa County Health Department reported on Wednesday five more human cases of West Nile virus."* produced a false negative, but when we removed the phrase *"on Wednesday"* all the dependencies were

correct and the sentence pair was found equivalent. The sentence pair became "Five more human cases of West Nile virus were reported by the Mesa County Health Department. *The Mesa County Health Department reported five more human cases of West Nile virus."*

Another problem occured when unexpected punctuation marks appeared in the middle of a sentence. One example was the sentence *"FBI agents arrested a former partner of Big Four accounting firm Ernst & Young ERNY.UL on criminal charges of obstructing federal investigations, U.S. officials said on Thursday."*. The parser interprets this as two sentences where *"ERNY."* is the end of the first sentence, and *"UL"* is the beginning of second one. When numbers with decimals appeared in a sentence the parser produced correct results as the point was not seen as a punctuation mark.

Another situation that caused false negatives was that in some sentences, dependencies were generated for the active sentence and some others for the passive sentence. This caused that in some cases there were remaining or "extra" dependencies for one or both sentences in the pair, causing the system to trigger non-equivalence for a pair of sentences. Two of the 15 false negatives found in this test were caused by this issue.

Table 5–5 shows the test results for Case 5.

Table 5–5: Case 5 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
55	51	0	4	4

In this case, one of the problems was that one of the template dependencies, the nominal subject dependency, *"nsubj"*, of the "will type" sentence, was incorrectly set by the system, causing one of the four false negatives found in this case. This occurred because there were two similar dependencies that were only different by their dependents. Another problem found was that the required controlling subject dependency *"xsubj"* was not generated in two sentence pairs. This problem was not caused by the system, but by the parser or the structure of the sentences. Two of the four false negatives were caused by this particular problem. The last problem found was that there was a remaining dependency in the non-template ones, causing the pair result non-equivalent as a false negative. The system did not give false positives in this case.

The system was also tested with homographs. The following pair of sentences was used for this purpose: *"The will of Will will be opened by Will's wife after Will is dead. The will of Will is going to be opened by Will's wife after Will is dead."* In this pair, the word "will" appeared with different meanings, but the system correctly classified them as equivalent.

Table 5–6 displays the results of testing the system for Case 6.

Table 5–6: Case 6 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
42	36	0	4	2

In this case, one of the false negatives occurred when the parser failed to identify the nominal subject dependency *"nsubj"* in the two sentences of the pair *"Bloomberg said on Wednesday that all 16 crew members survived and would be tested for drugs and alcohol. Bloomberg said on Wednesday that all 16 crew members survived and*

are going to be tested for drugs and alcohol.". The new version of the parser set this dependency correctly but since it was released in November of 2010, this version was not used since it would have given rise to significant changes in our code. The other false negative occurs because there was a remaining dependency in the non-template ones, as occured in Case 5. As this case is very similar to Case 5, it is not uncommon to have similar problems. The system did not give false positives in this case.

Table 5–7 shows the results for Case 7.

Table 5–7: Case 7 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
30	25	0	4	1

In this case, there was only one sentence pair which failed. The problem was that one of the template dependencies was not generated in any of the sentences of the pair, while one of the non-template dependencies generated was not equivalent between the pair. The sentence pair that failed was *"The center's president, Joseph Torsella, is struck on the head but is able to walk to an ambulance. The center's president, Joseph Torsella, is struck on the head but can walk to an ambulance."* Here, the template dependency *"nsubj"* was not generated in any sentence of the pair.

As previous cases, all real non-equivalent sentence pairs of this case were correctly identified causing no false positives.

Table 5–8 shows the results for Case 8.

Table 5–8: Case 8 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
26	17	0	2	7

Case 8 test results are shown in Table 5–8. As in Case 2, we used WordNet to get the antonyms and if the antonym was not in WordNet the system failed. An example of this was the antonym pair "die" and "live". Using the online dictionaries Thesaurus and Merriam-Webster we found that these words were antonyms, but in WordNet they did not appear as such. This WordNet limitation caused two of the seven false negatives. Another problem was that in two of the sentence pairs, the template dependencies were not correctly identified by the parser. Instead of the *"nsubj"* dependency, a generic dependency *"dep"* was generated in two of the cases. An example of this problem was the sentence pair *"State media said there were at least 770 dead and over 5,600 injured. State media said there were at least 770 not alive and over 5,600 injured."*, where in neither of the sentences the *"nsubj"* template dependency was not generated, but in the second sentence, a *"dep"* dependency was generated "in replacement" of the *"nsubj"*, *dep(alive-10, 770-8)*. Therefore, three of the seven sentences pairs were found not equivalent.

Another problem found was that if a sentence pair contained more than one negation, including the use of the adverb *"never"*, the system was not able to determine correctly the negation sentence and the sentence with the antonym, and it set both sentences as the negation sentence. Two of the seven false negatives found in this test were caused by this problem. As in previous cases, no false positives were found.

5.2.2 Combined cases results and discussion

In this section, the results of the combined cases are shown. From cases nine to thirteen are combinations using synonyms (Case 2), while Case 14 combines with contractions (Case 3).

Table 5–9 shows the results for Case 9.

Table 5–9: Case 9 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
12	5	0	2	5

In this case, all failed sentences had problems with the non-template dependencies. In these sentences, some of these dependencies were not equivalent between the sentences. If sentences failed for Case 4, the master case, the sentence pair was declared non-equivalent as expected from the way combined cases were constructed. If the sentence pair of the master case failed, the combined case must fail, even if all the synonyms used in the sentence pairs were found in WordNet.

As in previous cases, no false positives were found. All real non-equivalent sentence pairs of this case were correctly classified as non-equivalent.

Table 5–10: Case 10 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
14	11	0	3	0

Table 5–10 shows the results for Case 10. In this case, all tested sentences were classified correctly. All the tested sentences were simple future pairs and all synonyms used in them were found in WordNet.

Table 5–11: Case 11 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
14	8	0	2	4

Table 5–11 shows the results for Case 11. In this case, we had similar problems as in Case 2 . One of the problems was that two sentence pairs used verb synonyms and our system did not recognize them, as explained in Case 2. Therefore, the sentence pairs were found non-equivalent causing false negatives. Two of the four false negatives were caused by this problem . The other problem, related to Case 2, was that WordNet did not recognize two synonyms, causing a false negative. Another problem found was that the required controlling subject dependency *"xsubj"* was not generated, causing another false negative. This problem was similar to one of the problems found on Case 5. There was no problem related to Case 6 and there were no false positives in the tested sentences. The two sentence pairs found non equivalent, were true ones.

Table 5–12: Case 12 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
10	8	0	2	0

Table 5–12 presents the results for the combined Case 12. In this case, all sentences were correctly classified; there were neither false positives nor false negatives in the tested sentences. All synonyms used were found in WordNet. The non-equivalent sentence pairs of this case were classified correctly.

Table 5–13: Case 13 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
10	7	0	2	1

Table 5–13 presents the results for the combined Case 13. In this case there was only one false negative caused by the non-template dependencies. In the negation sentence, a passive auxiliary dependency was generated by the auxiliary *"was"*, while in the other sentence, this auxiliary generated a copula dependency. This makes the non-template dependencies not equivalent. In the other sentences, all synonyms used were found in WordNet, and all sentences were correctly classified.

Table 5–14: Case 14 Results

Total of sentence pairs	Classified as Equivalent		Classified as Non-equivalent	
	Success	False Positive	Success	False Negative
15	13	0	2	0

Table 5–13 presents the results for the combined Case 14. In this case all sentences were correctly classified, including those which were real non-equivalent. Of all sentence pairs used in this case, some of them were used in Case 5 and on Case 3 and they worked in both cases, so it was expected that they worked in this case as well. The other sentences belonged only to this case. The equivalent pairs were found correctly as equivalent of this case, while the non-equivalent pairs were found correctly as non-equivalent by our algorithm. Neither false positives nor false negatives were found in this case. The same way that Case 3 was combined with Case 5 in this case, it can be combined with other cases to create more combined cases, as done with Case 2.

5.2.3 Summary of results and discussion

In this section, the results of all cases are shown. Table 5–15 shows a summary of the results of the cases, while Table 5–16 shows the results of the totals of all the tests. In the same way, Figure 5–1 shows, the results of the tests.

Table 5–15 shows the results for each case, in terms of the total of sentence pairs that were tested in each case, the total of sentence pairs for which their equivalence was determined correctly either as semantically equivalent or non-equivalent, the total of sentence pairs for which their equivalence were determined incorrectly, and the percentage of correctly determined sentence pairs in each case.

Table 5–15 shows that four cases had perfect scores in percentage of successes, while two cases had near half the percentage. It is observed that six cases are in the range of 90 and 99 percent successful classification, and two cases are in the range of 70 and 79 percent.

One of the perfect score cases was Case 1, which is the most basic case, while two of them are combined cases using synonyms, and the remaining one is a combined case using contractions . The cases that had near the lowest success percentage were Case 4 and 9 which were related because Case 9 is a combined case of Case 4 and Case 2. The results of Case 9 were consistent with the results of Case 4. In all the sentence pairs tested from this corpus, no false positives were found.

Table 5–15: Summary of the results of the cases

Case	Total of sentence pairs	Equivalence successfully classified	Equivalence fail	Percentage of successful classification
1	33	33	0	100
2	60	54	6	90
3	98	97	1	98.9795918
4	32	17	15	53.125
5	55	51	4	92.7272727
6	42	40	2	95.2380952
7	30	29	1	96.6666667
8	26	19	7	73.0769231
9	12	7	5	58.3333333
10	14	14	0	100
11	14	10	4	71.4285714
12	10	10	0	100
13	10	9	1	90
14	15	15	0	100

Table 5–16 shows a summary of all the cases with the total of sentence pairs in the corpus, the total sentence pairs correctly and incorrectly classified, and the percentage of successful classification for the entire corpus. Table 5–16 shows that out of the 451 sentence pairs of the corpus, approximately 89.8% of them were correctly classified by our system.

Table 5–16: Overall cases summary

Total of sentence pairs	Equivalence succeed	Equivalence fail	Percentage of succeed
451	405	46	89.8004435

Figure 5–1 shows a graphic representation of the results of all cases, in terms of equivalent and non-equivalent sentences, false positives and false negatives.

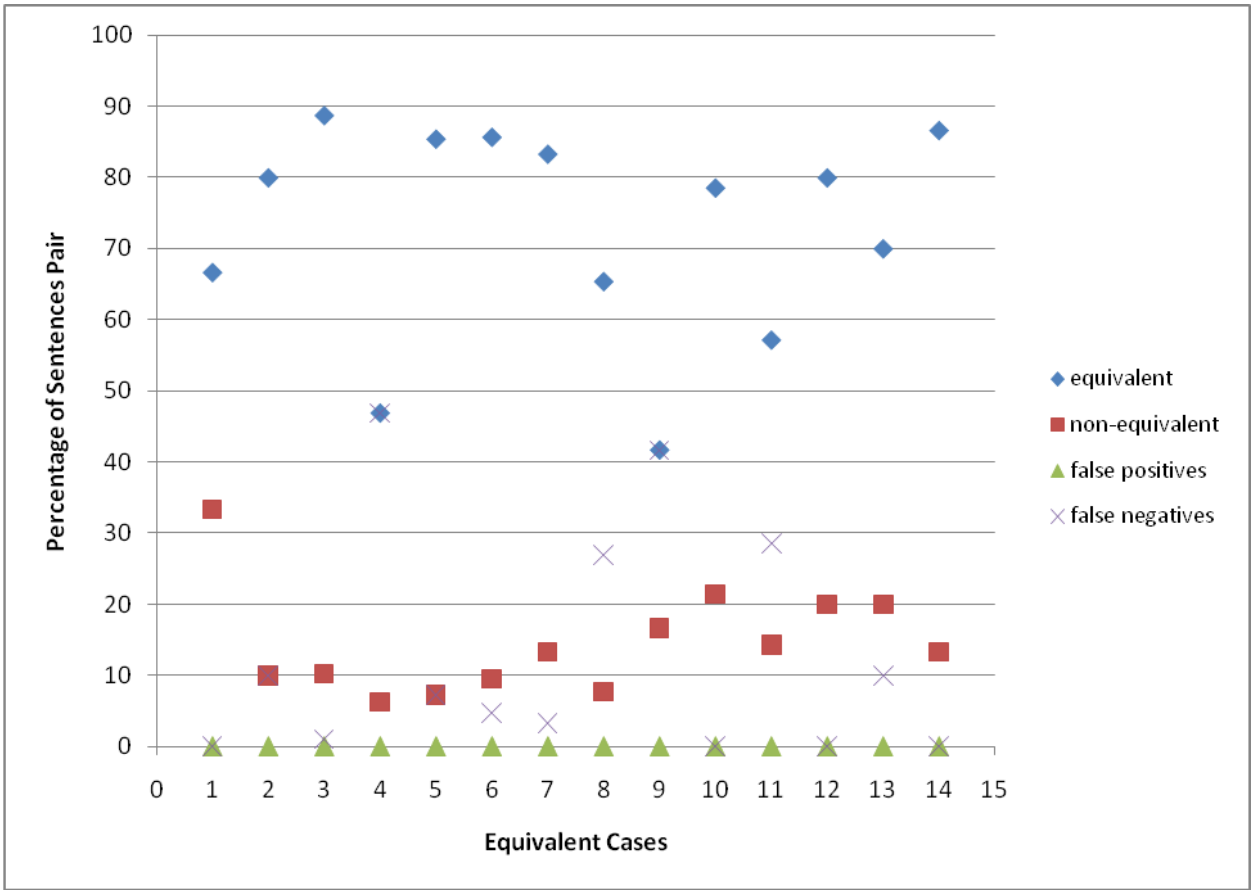

Figure 5–1: Graphic of results

5.3 Comparison with other systems

We found only one system that aimed at classifying equivalent sentences pairs. The system proposed by Newman et al [13] used semantic entailment to detect semantic equivalence in texts. The system used three corpora: the MSRP, a modified version of the MSRP, and the RTE. This system classified the sentence pairs as

equivalent or non equivalent and they rated the performance of the system based on the number of correctly and incorrectly classified sentence pairs. While their system obtained 63.94% to 74.00% of accuracy, our system achieved 89.8% of accuracy. It is important to note that they used entailment as a way to find equivalence and therefore, they used MSRP and RTE corpora without modifications. The fact that equivalence and entailment are two different notions led them to two consequences:

- to mistakenly classify sentences as equivalent, and
- to reduce the rate of correct classification

We can conclude that the methods employed in our system are more effective and specific than the system described in [13].

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

In the previous chapters, the research, methodology and tests applied to the algorithms for determining semantic equivalence between pair of sentences based on 14 different semantic equivalence cases were presented. We now proceed to present the conclusions of our work.

6.1 Conclusions

Semantics is the study of meaning in language [8]. The semantic equivalence between sentences determines if the sentences share the same meaning. It can be defined using entailment, since two sentences are logically equivalent if they entail each other.

As stated before, shallow semantics is the kind of semantics based only on grammatical information without resorting to causal chains or any other abstract knowledge, while deep semantics was defined as the kind of semantics which needs in-depth knowledge of the subject. Shallow semantics was the semantic approach selected to design the method to find semantic equivalence between two sentences. This semantic approach covers a wide variety of subject domains without requiring specialized knowledge representation. Shallow semantics is based on grammatical information which can be used to detect patterns which help to define the semantic equivalence algorithms used for each semantic equivalence case studied.

The Stanford Natural Language Parser was used to extract the grammatical information from the sentences, while WordNet was the lexical database used to find synonyms and antonyms in our system. The Stanford Natural Language Parser is a statistical, or probabilistic, parser which produces the grammatical information of a text in different formats. Some of them were extracting the Part-Of-Speech word tags, generating a tree structure of the sentences, and generating type dependencies. On the other hand, WordNet is a lexical database composed of synsets, which are synonym sets, and are logical groups where the information is organized.

The system consisted of the equivalence cases algorithms, the Stanford parser to get the grammatical information of the sentences, the WordNet database to work with the synonyms and antonyms, and the CSV files which work as the database that contained the equivalence in verbs, pronouns, and contractions used by our algorithms. The algorithms of the system were developed using the Java language, and the database was simulated using SuperCSV. The combined cases were developed using Java inheritance and interfaces. Semantic equivalence cases were developed using the grammatical information obtained from the Stanford Natural Language parser, and looking for patterns observed in each case.

To test the system a corpus of semantic equivalent sentence pairs was developed by modifying the Microsoft Research Paraphrase corpus, MSRP. For each case, there was an amount of sentence pairs. The system tests were performed either reading the sentence pairs directly from the corpus file or entering a pair by console. Then, the system classified the sentence pair as semantic equivalent or not in one of the equivalence cases and the output produced was displayed on the console.

The results obtained in our tests show that our system is reliable overall. Only cases involving active and passive voice, *i.e.*, Case Four and Case Nine produced results that were not favorable. These cases need more research using deep semantics.

There are other systems with similar tasks to our system, but we found only one system which was more related to our work. The system proposed by [13] tries to detect semantic equivalence in texts using an entailment approach based on the premise that, although semantic entailment is not the same as semantic equivalence, they are closely related. This approach differs from ours because our system uses pre-defined cases to find equivalence between sentences. In terms of the results, our system obtained more accurate results than their system, thus we can conclude that the methods employed in our system, by being more specific, result more effective than the methods of their system.

One similarity that the system shares with our system is that they used WordNet as a tool to find synonyms in sentences, and the MSRP corpus to perform the tests. In addition to the MSRP corpus, they used the Pascal RTE challenge corpus to perform the tests. This corpus was not available at the time of this thesis.

The following are the conclusions of this thesis:

- A case-based algorithm for detecting semantic equivalence among sentences in English was developed using shallow semantics. The use of shallow semantics makes the approach domain independent.
- The use of shallow semantics allowed detecting equivalence without the need for knowledge representation beyond the grammatical information in the sentences.

- Fourteen equivalence cases were defined, of which eight are base cases and six are combinations of these base cases.
- The main tools used in this thesis were the Stanford natural language parser, which uses dependency grammar, Wordnet for identifying synonyms and antonyms and CSV files which are used as a database of verb conjugation, pronoun and contraction equivalences.
- A corpus of 451 pairs of equivalent sentences was developed; this is a modification of the MSRP corpus.
- The system developed in this thesis achieved a rate of 89.80% of successful equivalence detection, which improves the percentage rate reported in the literature, which achieves between 63.94% to 74.00% success rate.

6.2 Future Work

There is not much information about semantic equivalence because both semantic equivalence and semantic entailment are understudied fields in natural language processing [12]. This limitation allows for more academic research.

As future work, we propose to explore deep semantics to handle more cases, have more precision in the results, and investigate semantic entailment as a generalization of semantic equivalence. The inclusion of deep semantics would solve some ambiguities, a problem that cannot be addressed using shallow semantics.

A great improvement to this work is adding another semantic equivalence case which was discussed during the development of this thesis but due to the scope of this work, it was not possible to design and implement. This case was called *Order changed sentences*. This case was supposed to be in charge of determining semantic

equivalence among two sentences which words are written in different order, maintaining a correct grammar, without altering the sentence meaning. There is not a defined pattern to write sentences with the words in different order which works in every sentence; a sentence can be written in many ways, meaning the same. For this reason, shallow semantics is not appropriate to handle this case. The development of this case is possible with the use of deep semantics. Deep semantics would help making possible disambiguation in some cases and doing the connections that determine the equivalence of a sentence pair. Examples of this type of sentences are {"Sam, my brother, eats red meat.", "My brother Sam eats red meat."}, {"I love cheese and chocolate.", "I love chocolate and cheese."}, {"She can eat meat, chicken or pork.", "She can eat pork, meat, or chicken."}.

The *Order changed sentences* case has some subcases. We called one of them *The Yoda Case*. We name this case because in this case the sentences are written the way Yoda [1] speaks. An example of semantic equivalent pair of sentences in *Yoda Case* subcase is {"You must study for the exam.", "Study for the exam, you must."}.

One feature that can be extended to improve the system is the addition of more types of synonyms, such as verbs, adjectives and adverbs. Also, the system can be improved by reading more than one pair of sentences at a time, and showing the results of each sentence pair separately.

Another improvement to the system is the addition of more combined cases. But, the best improvement is combining the cases using multiple combinations.

[1] Yoda is the Star Wars movie character (TM & ©Lucasfilm Ltd)

The combination of cases like *Simple Future* with *Sentences using Active and Passive Voice*, *Negation with Antonym with Sentences using "can" and "be able to" verb forms*, or *Future in the Past with Negation with Antonym*, to mention just a few, greatly helps to improve this system. In addition, the combination of cases using more than two cases would improve the system, making it more practical, efficient, and complete.

Natural language semantic analysis is a complex and challenging field of artificial intelligence. The applications are numerous and would certainly improve our interaction with computers, information, knowledge and even with other people. This thesis looked into one problem and one approach, and it has revealed that the work ahead is of great interest and utility academically, or businesses and the people in general as we move from the information into the knowledge age, in computational semantics will allow a more natural communication between humans and computers.

APPENDICES

APPENDIX A
ENGLISH GRAMMAR

Below, we present a brief summary of English grammar that is essential to the understanding of the work in this thesis.

A sentence is composed of phrases, and the phrases are composed of words. These are elements of grammar. Grammar is defined as a set of rules used to combine words to form phrases, sentences, and other larger units in a defined language. Grammar is also known as syntax and is the central component of a language [6].

In this thesis, are work with English grammar.

A.1 Parts of speech

Parts of speech are grammatical categories which classify the words according to their use in sentences. The traditional parts of speech are eight: noun, pronoun, verb, adjective, adverb, preposition, conjunction, and interjection. A brief description of each one follows.

A.1.1 Noun

A noun refers to the name of a person, place or thing. Nouns can have number (singular or plural), and case (subjective/objective or possessive). Nouns can be common or proper [6].

A.1.2 Pronoun

A pronoun substitutes a noun and there are seven types: personal, demonstrative, interrogative, indefinite, relative, reflexive, and reciprocal. The personal pronouns refer to specific objects and have the inflection of person: first, second and third person [6]. Demonstrative pronouns point out specific things. Interrogative pronouns introduce questions in a direct or indirect form. Indefinite pronouns refer to something without specifying it. Relative pronouns introduce an adjective clause. Reflexive pronouns refer to the same noun or pronoun as the subject, acting on itself. Reciprocal pronouns refer to a reciprocal relationship of two or more subjects.

A.1.3 Verb

The verb is the part of speech that describes the action or the state of being of the sentence. The verbs can be conjugated in different ways: by person, by number, by tense and by mood. In the person conjugation, the first, second and third person are the categories. In the conjugation by number, there are singular and plural. With regard to tense, there are past and present, and in the mood way, there are indicative, subjunctive and imperative.

There are four basic forms of verbs: infinitive or base form, finite, present participle, and past participle. Of these categories, the infinitives, past participles and present participles are called verbals, because they are often used with auxiliaries. The infinitive is the basic form of the verb, and corresponds to the present tense preceded by the preposition “to”. The finite verb makes a statement about a subject and can be conjugated by person, number, tense and mood. The finite verbs are very important in clauses; every clause needs one. Finite verbs are usually in present or past tense. The present participle usually expresses ongoing, repeated or habitual action, and usually ends in “ing”. Sometimes, these verbs are used as

adjectives, nouns and as part of a periphrastic verb. The past participle verbs are used for passive sentences. Table A–1 illustrates how a regular and an irregular verb are conjugated in these forms.

Table A–1: Regular and irregular English verb conjugation

Infinitive	Finite		Present participle	Past participle
	Present	Past		
Laugh	Laughs	Laughed	Laughing	Laughed
Speak	Speaks	Spoke	Speaking	Spoken

Of these categories, the infinitive, the past tense, and the past participle are considered the principal parts of a verb, because they are the only forms that must be memorized. The present tense is basically the infinitive with a -s termination and similarly, the present participle is the infinitive with an -ing termination [6].

In addition to that, a verb must have the same number and person as the subject [6]. This constraint applies to the first verb, regardless of whether it is the main or an auxiliary verb, except modal auxiliaries that do not make distinctions in number or person [6].

Verbs have two voices: active and passive. The passive verb has one auxiliary more than the active one, and it is inflected in past participle followed by the preposition “by”. The passive voice is a way of paraphrasing a sentence, making the subject not to refer to the thing responsible for the action [6].

As previously stated, verbs have only present and past tenses. Future tense is expressed using the present tense or using a future auxiliary verb. An example of expressing future using present tense is the sentence “My sister arrives tomorrow.”, while an example expressing future using an auxiliary verb is “My sister will arrive

tomorrow."

A.1.4 Adjective

The function of the adjective is to modify the noun. In English, adjectives are normally placed before the noun of the sentence [48]. In the example sentence "The blue car is mine.", the word "blue" modifies the noun "car", then "blue" is the adjective.

A.1.5 Adverb

The adverb modifies an adjective, verb and other adverb. The most known adverbs are those that end in -ly. The conjunctive adverbs are easily confused with conjunctions because their use and meaning are similar to them.

A.1.6 Preposition

The preposition introduces a prepositional phrase, a noun phrase or a pronoun [48]. Some preposition examples are: about, after, before, between, but, by, during, for, from, in, to, with [6].

A.1.7 Conjunction

The conjunction joins elements in sentences or phrases. They come in three flavors: coordinating, subordinating and correlative conjunctions. Coordinating conjunctions join words in a direct way, using words like "and", "or", "but", among others. Subordinating conjunctions join the subordinate and the principal clause using words like when, where, although, among others. The correlative conjunctions join words in pairs; either ... or ..., both ... and ... are examples.

A.1.8 Interjection

The interjection is a word which expresses any kind of emotion, an exclamation. Common examples of interjections are: oh, ah, well, ouch, whoa, among others.

A.2 Phrases

A phrase is the part of a sentence that contains a coherent group of words but contains either a subject or a verb. It can be defined as a group of words which compose a simple clause [48]. There are five types of phrases: noun, verb, adjective, adverb, and, prepositional phrases [6]. Noun phrases have a noun or a pronoun as the main word, and usually determiners introduce them [6]. Verb phrases have a verb as the main word, and are composed of one main verb and optional auxiliary verbs [6]. Adjective phrases have an adjective as the main word and are composed of the adjective and optional modifiers [6]. Adjective phrases can work as a pre-modifier in a noun phrase, as a subject complement, as an object complement, and as post-modifier in a noun phrase [6]. Adverb phrases have an adverb as the main word and are composed of the adverb and optional modifiers. An adverb phrase can modify an adjective phrase or another adverb phrase, and also can work as an adverbial in a sentence structure [6]. Prepositional phrases are composed of a preposition and a complement. The complement is usually a noun phrase, but it may also be a nominal relative clause or an -ing clause. Prepositional phrases can work in three ways: as a post-modifier of a noun, as a post-modifier of an adjective, or as an adverbial [6].

Phrases are lower in grammatical category than clauses [48].

A.3 Clauses

Clauses are simple sentences that state one predicate with its arguments. A clause must have a word expressing the predicate (verb) and a phrase expressing

the argument number and type for the predicate (subject). One type of clause is the subordinate clause. It is usually introduced by a subordinator, and has three subcategories: nominal, modifier, and adverbial. Nominal clauses, also known as noun clauses, have similar functions as noun phrases. Within nominal clauses, nominal relative clauses exist, which are introduced by a nominal relative pronoun. Modifier clauses have similar functions as modifiers in phrases. Examples of modifier clauses are the relative clause, which form part of the noun phrase, and the comparative clause. Adverbial clauses have similar functions as the adverbial in sentences.

A.4 Sentences

The sentence is the largest unit of grammar [6]. It is composed of one or more clauses; a single sentence has one clause while a multiple sentence has more than one [6]. There are four main types of sentences: declaratives, interrogatives, imperatives, and exclamatives. Declarative sentences are those which express information or a statement; interrogative sentences ask a question; imperative sentences give a command; and exclamative sentences express an emotion.

Sentences can be written in positive or negative form. Positive sentences may have an auxiliary verb in positive form. If an auxiliary verb is present, a positive sentence can be transformed into a negative one by adding a negative word after the auxiliary verb. Negative words can be: “no”, “not”, the contraction “n’t”, the adverb “never”, the pronoun “nobody”, among others. In addition to these forms, a sentence can be written in active or passive voice. In these two forms, the verb has different conjugations and the order of the words is different [6]. For the active sentences, the verb is written in simple present or simple past.

For the active sentences the subject and the object are inverted in position with respect to the passive sentences. In the same way, the active sentence verb is written in present, past or future tense, while the passive sentence verb is written in past participle tense preceded by an auxiliary verb which must have the same tense as the active sentence verb and followed by the "by" preposition [6]. The following pair of sentence illustrates this:

Active: "Charles Dickens wrote many novels."
Passive: "Many novels were written by Charles Dickens."

Sentences have some elements. A sentence consists of two main parts: the subject and the predicate [6]. The subject usually is placed before the verb in a declarative sentence, before the operator in interrogatives, and is absent in imperatives and exclamatives [6]. One way to identify the subject in a declarative sentence is transforming the sentence in a what or who question; the answer to the question is the subject of the sentence. The predicate consist of the main verb and the other parts that are not the subject. The predicate must contain at least one verb, and can be composed of only one verb. The verb is the most important component of a predicate, and in fact, is the more important component of a sentence. A sentence can contain more than one verb. Usually there is a main verb and other verbs, which help the main verb. These verbs are placed before the main verb and are called auxiliary verbs [6]. The verb is easily identified because it changes its form or has auxiliaries which express the time or the attitude of the sentence [6].

The operator is one component of the verb that is part of the predicate and has an important function in a sentence. The operator is the first or only auxiliary verb

of the sentence [6].

There are three types of verbs: transitives, intransitives and copula. Transitive verbs are those in which the subject of the sentence performs an action that affects some person or thing. The direct object is what is directly affected by the verb or action. In a declarative sentence, the direct object can be identified transforming the sentence in a what or who question, with the operator and the subject following the question word (i.e: what or who); the answer of the question is the direct object of the sentence [6]. The indirect object is what is indirectly affected by the verb or action, and usually is a recipient of something. Usually, it is introduced by words like "to" or "for", and comes after the verb and the direct object [6]. Intransitive verbs are those which neither need a direct object nor another element to complete the sentence [6].

The copula is like a linking verb, a verb used to link the subject with the predicate of a sentence. The verb "to be" is usually a copular verb. They are usually followed by a subject complement [6].

Adverbials are the optional elements that can be added to a sentence, except the adverbial complements. Adverbials transmit more information about the situation the sentence expresses and are different to adverbs [6]. The first ones are similar to the subject; they are sentence constituents, while the second ones are similar to nouns; they are words. Adverbial complements transmit the same information as adverbials but complete the information of the main verb, reason why they are required [6].

In summary, a sentence is composed of the subject, the verb, the object and the complement. There are two types of objects: the direct and the indirect object. There are three types of complement: the subject complement, the object complement, and the adverbial complement.

REFERENCE LIST

[1] S.J. Russell, P. Norvig, J.F. Candy, J.M. Malik, and D.D. Edwards. *Artificial intelligence: a modern approach*. Prentice hall, third edition, 2010.

[2] Wolfgang Wahlster, Henry Lieberman, and James Hendler. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. The MIT Press, November 2002.

[3] Dave Raggett. W3C: Getting started with HTML. http://www.w3.org/MarkUp/Guide.

[4] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web: Scientific American. *Scientific American*, May 2001.

[5] J. Feng, Y. Zhou, and T. Martin. Recognizing Textual Entailment Based on WordNet. In *Second International Symposium on Intelligent Information Technology Application*, pages 27–31. IEEE, 2008.

[6] S. Greenbaum and G. Nelson. *An introduction to English grammar*. Pearson Education, second edition, 2002.

[7] M. Stanojevic and S. Vranes. A Natural Language Processing for Semantic Web Services. In *Computer as a Tool, 2005. EUROCON 2005. The International Conference on*, volume 1, pages 229–232. IEEE.

[8] J.R. Hurford, B. Heasley, and M.B. Smith. *Semantics: a coursebook*. Cambridge University Press, second edition, 2007.

[9] Britannica Online. Encyclopædia Britannica Online: meaning. http://www.britannica.com/EBchecked/topic/371586/meaning, 2009.

[10] H. Marmanis and D. Babenko. *Algorithms of the Intelligent Web*. Manning Publications Co., 2009.

[11] N. Shadbolt, W. Hall, and T. Berners-Lee. The Semantic Web Revisited. *Intelligent Systems, IEEE*, 21(3):96 –101, jan.-feb. 2006.

[12] P. Achananuparp, X. Hu, and C. Yang. Addressing the Variability of Natural Language Expression in Sentence Similarity with Semantic Structure of the Sentences. *Advances in Knowledge Discovery and Data Mining*, pages 548–555, 2009.

[13] E. Newman, J. Carthy, J. Dunnion, and N. Stokes. Identifying semantic equivalence for multi-document summarisation. *Artificial Intelligence Review*, 25(1):55–65, 2006.

[14] M. Tatu and D. Moldovan. A semantic approach to recognizing textual entailment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 371–378. Association for Computational Linguistics, 2005.

[15] R. de Salvo Braz, R. Girju, V. Punyakanok, D. Roth, and M. Sammons. An inference model for semantic entailment in natural language. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 20, page 1043. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[16] The Stanford Natural Language Processing Group. The Stanford Parser: A statistical parser. http://nlp.stanford.edu/software/lex-parser.shtml.

[17] Marie catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *In LREC 2006*, 2006.

[18] Michael Sipser. *Introduction to the Theory of Computation*. PWS Pub., 2 edition, 2006.

[19] Gerard Salton. *Automatic Text Processing*. Addison-Wesley Publishing Company, 1989.

[20] Michael L. Scott. *Programming Language Pragmatics*. Morgan Kaufmann Publishers Inc., 3rd edition, 2009.

[21] D. Rusu, L. Dali, B. Fortuna, M. Grobelnik, and D. Mladenic. Triplet Extraction From Sentences. *Proceedings of the 10th International Multiconference "Information Society - IS 2007"*, A:218–222, 2007.

[22] Marie-Catherine de Marneffe and Christopher D. Manning. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8. Association for Computational Linguistics, 2008.

[23] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31, 2010.

[24] Jomol P Mathew, Barry S Taylor, Gary D Bader, Saiju Pyarajan, Marco Antoniotti, Arul M Chinnaiyan, Chris Sander, Steven J Burakoff, and Bud Mishra. From Bytes to Bedside: Data Integration and Computational Biology for Translational Cancer Research. *PLoS Comput Biol*, 3:e12, 02 2007.

[25] Pan Du, Gang Feng, Jared Flatow, Jie Song, Michelle Holko, Warren A. Kibbe, and Simon M. Lin. From disease ontology to disease-ontology lite: statistical methods to adapt a general-purpose ontology for the test of gene-ontology associations. *Bioinformatics*, 25(12):i63–i68, June 2009.

[26] Nicola Henze, Peter Dolog, and Wolfgang Nejdl. Reasoning and ontologies for personalized e-learning in the semantic web. *Educational Technology & Society*, 7:82–97, 2004.

[27] Serge Linckels and Christoph Meinel. Automatic Interpretation of Natural Language for a Multimedia E-learning Tool. In Nora Koch, Piero Fraternali, and Martin Wirsing, editors, *Web Engineering*, volume 3140 of *Lecture Notes*

in Computer Science, pages 778–778. Springer Berlin / Heidelberg, 2004.

[28] Ping Liu, Yan Ye, and Kan Liu. Building a Semantic Repository of Academic Experts. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, oct. 2008.

[29] Xin Xin, Juanzi Li, and Jie Tang. Enhancing Semantic Web by Semantic Annotation: Experiences in Building an Automatic Conference Calendar. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, WI '07. IEEE Computer Society, 2007.

[30] Raphaël Troncy, Jacco van Ossenbruggen, Jeff Z Pan, and Giorgos Stamou. W3C: Image Annotation on the Semantic Web. http://www.w3.org/2005/Incubator/mmsem/XGR-image-annotation-20070814.

[31] Evgeniy Gabrilovich and Shaul Markovitch. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research*, 34, March 2009.

[32] Baowen Xu, Dazhou Kang, Jianjiang Lu, Peng Wang, and Yanhui Li. Equivalent Individuals on the Semantic Web. In *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration, IRI - 2004*, pages 253–258. IEEE Systems, Man, and Cybernetics Society, 2004.

[33] T. Segaran, C. Evans, and J. Taylor. *Programming the semantic web*. O'Reilly Media, first edition, 2009.

[34] Joseph C. Giarratano and Gary D. Riley. *Expert Systems: Principles and Programming*. Brooks/Cole Publishing Co., fourth edition, 2005.

[35] Sanda Harabagiu, Cosmin Adrian Bejan, and Paul Morarescu. Shallow Semantics for Relation Extraction. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05. Morgan Kaufmann Publishers Inc., 2005.

[36] X.Y. Liu, Y.M. Zhou, and R.S. Zheng. Measuring Semantic Similarity within Sentences. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 5, pages 2558–2562. IEEE, 2008.

[37] A. Andreevskaia, Z. Li, and S. Bergler. Shallow Semantics for Textual Entailment Determination. 2009.

[38] *Techniques for Recognizing Textual Entailment and Semantic Equivalence*, volume 4177. Springer, 10/2006 2006.

[39] Chris Quirk, Chris Brockett, and William Dolan. Monolingual Machine Translation for Paraphrase Generation. In *In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 142–149, 2004.

[40] Microsoft Natural Language Processing Group. Microsoft Research Paraphrase Corpus. http://research.microsoft.com/en-us/downloads/607d14d9-20cd-47e3-85bc-a2f65cd28042/.

[41] M.C. De Marneffe and C.D. Manning. *Stanford typed dependencies manual*, 2010.

[42] L. Tesnière. *Eléments de syntaxe structurale*. Libraire C. Klincksieck, 1959.

[43] WordNet online documentation. Glossary of WordNet terms. http://wordnet.princeton.edu/wordnet/man/wngloss.7WN.html.

[44] J.S. DeCARRICO. Tense, aspect, and time in the english modality system. *TESOL Quarterly*, pages 665–682, 1986.

[45] J.R. Martin. Linguistics and the consumer: The practice of theory. *Linguistics and education*, 9:411–448, 1997.

[46] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[47] WordNet. WordNet Web Site. http://wordnet.princeton.edu.

[48] P. Kroeger. *Analyzing grammar: An introduction*. Cambridge Univ Pr, first edition, 2005.

COMPUTERIZED DETECTION OF SEMANTIC EQUIVALENCE AMONG SENTENCES IN NATURAL LANGUAGE

Celibette Michelle Ossorio Laracuente
Department of Electrical and Computer Engineering
Chair: José Fernando Vega-Riveros
Degree: Master of Science
Graduation Date: December 2011