UNIVERSITY OF BIRMINGHAM

Automatic Documents Summarization Using Ontology based Methodologies

Abdullah Bawakid

Thesis submitted for the degree of
Doctor of Philosophy

School of Electronic, Electrical and Computer Engineering
College of Engineering and Physical Sciences
University of Birmingham

ABSTRACT

Automatic summarization is the process of creating a condensed form of a document or several documents. When humans summarize a document they usually read the text first, understand it then attempt to write a summary. In essence, these processes require at least some basic level of background knowledge by the reader. At the very least, the human would have to understand the Natural Language that the text is written in. In this thesis, an attempt is made to bridge the gap of machines' understanding by proposing a framework backed with knowledge repositories constructed by humans and containing real human concepts.

I use WordNet, a hierarchically-structured repository that was created by linguistic experts and is rich in its explicitly defined lexical relations. With WordNet, algorithms for computing the semantic similarity between terms are proposed and implemented. The relationship between terms, and a composite of terms, is quantified and weighted through new algorithms allowing for grouping the terms, phrases and sentences based on the semantic meaning they carry. These algorithms are especially useful when applied to the application of Automatic Documents Summarization as shown with the obtained evaluation results. Several novel methods are also adapted to enhance the diversity and reduce redundancy in the generated summaries.

I also use Wikipedia, the largest encyclopaedia to date. Because of its openness and structure, three problems had to be handled: Extracting knowledge and features from Wikipedia, enriching the representation of text documents with the extracted features, and using them in the application of Automatic Summarization. First, I show how the structure and content of Wikipedia can be used to build vectors representing human concepts. Second, I illustrate how these vectors can be mapped to text documents and how the semantic relatedness between text fragments is computed. Third, I describe a summarizer I built which utilizes the extracted features from Wikipedia and present its performance.

I demonstrate how the Wikipedia-extracted features can be adapted in applications other than Automatic Summarization such as Word Sense Disambiguation and Automatic Classification. A description for the implemented system and the algorithms used is provided in this thesis along with an evaluation.

DECLARATION

I confirm that this is my own work and that the use of all material from other sources has been properly and fully acknowledged.

Abdullah Bawakid

ACKNOWLEDGMENTS

I am extremely grateful to my supervisor Dr. Mourad Oussalah for investing so much time with me and the guidance he afforded me during my research. He gave me plenty of freedom to explore directions I was interested in. He has continued to be encouraging, patient and insightful. I thank also my backup supervisor Dr. Chris Baber for his support and commitment.

I have received partial support funds from the University of Birmingham. Thank you for your generous support.

I am very grateful to my parents, wife and family for their faith, support and patience.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

20N	20 Newsgroups
ADS	Automatic Documents Summarization
ANNIE	A Nearly New IE
BE	Basic Elements
BLEU	Bilingual Language Evaluation Understudy
BOW	Bag of Words
CREOLE	Collection of REusable Objects for Language Engineering
CT	Context Terms
DM	Data Mining
DUC	Document Understanding Conference
EM	Exact-Match Concepts
FG	Features Generator
GATE	General Architecture of Text Engineer
HTML	Hyper Text Markup Language
IC	Information Content
IDF	Inverse Document Frequency
IR	Information Retrieval
JCn	Jiang similarity metric
LE	Language Engineering
LSA	Latent Semantic Analysis
NE	Named Entity
NIST	National Institute of Standards and Technology
ODP	Open Directory Project
PJWSL	Pure Java WordNet Similarity Library
POS	Part-of-Speech
PR	Processing Resource
ROUGE	Recall Oriented Understudy for Gisting Evaluation
SSM	Sentences Simplification Module
SCU	Summaries Content Units
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TAC	Text Analysis Conference
TD	Test Document
TDS	Text Documents Summarization
TF	Term Frequency
TFIDF	Term Frequency , Inverse Document Frequency
TU	Text Unit
WSD	Word Sense Disambiguation
XML	eXtensible Markup Language

Chapter 1

Introduction

Automatic summarization is the process of creating a condensed form of a document or several documents. The documents can be text, video, speech, graphs, or any combination of these types. A summary presents the most significant parts of the document. It can be important facts or a shortened version of the details and descriptions present in the original document. Also, a summary can be tailored to present the user's specific requests and needs.

In this thesis, I describe the work completed for automatic summarization targeting documents of type text and taking into account the user's specific queries and requests. I explore variations of the system creating single and multi-documents summaries. I examine the new summarization methods implemented that would take advantage of the semantics present within the original documents and utilize external repositories for better semantic analysis.

1.1 Motivation

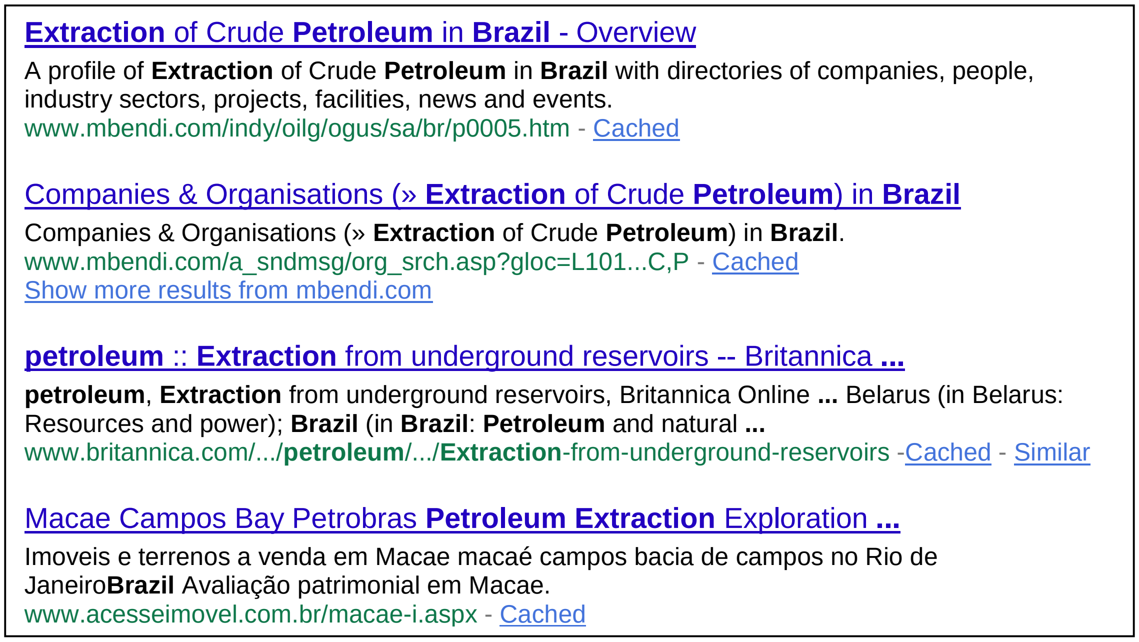
Recent expansion of the World Wide Web and the continuous size increase of affordable media storage devices have allowed for raw data to be available in very large quantities. With the abundance of data, comes the challenge of smart analysis and filtering which has to be applied to the data to reach the user's specific needs. This is especially apparent since no one is able to browse through all of the available data except for a small portion covering his needs.

Data Mining (DM) and Information Retrieval (IR) are two fields which have focused on the analysis of large amounts of data and extraction of high quality information relevant to the user's needs. With Data Mining, patterns and trends are typically detected within text to help identify and form interesting and important information. The field of Information Retrieval searches documents, data within documents and their metadata to help find important and relevant information. An overlap between the two fields is apparent, especially when looking at the sub areas which are covered by both. Among these areas are searching, as performed by internet users with web search engines; Sentiment Analysis, which attempts to determine the attitude of a person towards a subject; Text Categorization, which labels documents using previously-defined categories; Topics Extractions, which aims to extract the main topics mentioned in a document; Documents Clustering, which groups documents into a list of meaningful categories; and Automatic Documents Summarization, which targets the production of summaries meeting the user's needs.

Automatic Documents Summarization (ADS) is defined as the act of automatically creating a summary that briefly and succinctly presents the important information existing within the original documents. A summary can be generated for a single original document or a group of documents. The first is called a single-document summary while the later is a multi-document summary. Summaries can also be generic summaries or query-specific summaries depending on the user needs. The documents to be summarized can be of multimedia type, text, or both. Systems that focus on Text Documents Summarization (TDS) often involve subtasks borrowed from the Natural Language Processing field such as Text Parsing, Natural Language Understanding, Coreference Resolution, and Anaphor Resolution. Thus, TDS can be viewed as a subfield of Natural

Language Processing, which also overlaps with DM and IR. The focus in this thesis is on TDS.

To illustrate the need for a good TDS system, I start by giving an example. Suppose somebody who is looking into countries with emerging markets to invest in is investigating the economy of Brazil. In particular, he would like to learn more about its oil industry and its latest developments and discoveries in the country. An easy way for obtaining such info would be the use of a web search engine. Suppose that this person used the keywords "Petroleum Extraction Brazil" in the commonly used search engine Google. The result of this query could return the top four webpage links illustrated in Figure 1.1. Along with each link, Google provides a snippet summary taken from the original webpage.



Extraction of Crude **Petroleum** in **Brazil** - Overview
A profile of **Extraction** of Crude **Petroleum** in **Brazil** with directories of companies, people, industry sectors, projects, facilities, news and events.
www.mbendi.com/indy/oilg/ogus/sa/br/p0005.htm - Cached

Companies & Organisations (» **Extraction** of Crude **Petroleum**) in **Brazil**
Companies & Organisations (» **Extraction** of Crude **Petroleum**) in **Brazil**.
www.mbendi.com/a_sndmsg/org_srch.asp?gloc=L101...C,P - Cached
Show more results from mbendi.com

petroleum :: **Extraction** from underground reservoirs -- Britannica **...**
petroleum, **Extraction** from underground reservoirs, Britannica Online **...** Belarus (in Belarus: Resources and power); **Brazil** (in **Brazil**: **Petroleum** and natural **...**
www.britannica.com/.../**petroleum**/.../**Extraction**-from-underground-reservoirs -Cached - Similar

Macae Campos Bay Petrobras **Petroleum Extraction** Exploration **...**
Imoveis e terrenos a venda em Macae macaé campos bacia de campos no Rio de Janeiro**Brazil** Avaliação patrimonial em Macae.
www.acesseimovel.com.br/macae-i.aspx - Cached

Figure 1.1: Top 4 results for search query "Petroleum Extraction Brazil" as obtained from Google

Assuming that the rank of a result indicates its importance in relation to the user query, I examine the webpages of the results illustrated in the top four links. When examining each link, I make use of the norm that important information in articles is usually presented first to the reader. Therefore, I pay particular attention to the first four sentences of every webpage I examine. Starting with the first link, the sentences are displayed in Figure 1.2. One can notice that the article provides an overview of the oil reserve and production in Brazil.

Extraction of Crude Petroleum in Brazil
- According to the 2010 BP Statistical Energy Survey, Brazil had proved oil reserves of 12.856 billion barrels at the end of 2009 or 0.96 % of the world's reserves.
- Brazil produced an average of 2029 thousand barrels of crude oil per day in 2009, 2.62% of the world total and a change of 7.1 % compared to 2008.
- Brazil and Venezuela accounted for more than 60% of South America's oil production.
- Total oil production (including crude, natural gas liquids, ethanol and refinery gain) has been rising steadily since the early 1990s, with an average of 1.88 mmbd in 2003.

Figure 1.2: The first four sentences of the article ranked first in the search result for the second query

The second link points to a webpage containing merely a map showing the locations of some of the major companies in Brazil. The page has multiple random appearances of the keywords used in the query, and this may have affected its rank in the search results.

The third link points to a webpage giving a background about oil alternatives in the past and how its use for illumination triggered the different methods of extraction in the previous centuries. Figure 1.3 gives an overview of the top sentences in the article. It is clear that the webpage does not discuss exactly what the query is asking for, even though it has links to other pages discussing Petrobras; a major government-owned company for extracting and refining petroleum in Brazil.

Petroleum: Extraction from the underground reservoirs
- Until the beginning of the 19th century, illumination in the United States and in many other countries was little improved over that known by the early Greeks and Romans.
- The need for better illumination that accompanied the increasing development of urban centres made it necessary to search for new sources of oil, especially since whales, which had long provided fuel for lamps, were becoming harder and harder to find.
- By the mid-19th century kerosene, or coal oil, derived from coal was in common use in both North America and Europe
- The Industrial Revolution brought on an ever-growing demand for a cheaper and more convenient source of lubricants as well as illuminating oil.

Figure 1.3: The sentences extracted from the webpage titled "Petroleum: Extraction from the underground reservoirs"

The fourth link in the search result points to a webpage promoting for a Brazilian city called Macae. The city contains many facilities owned by the oil company Petrobras. The first four sentences are shown in Figure 1.4.

Macae Campos Bay Petrobras Petroleum Extraction Exploration
- Macae has a rapidly growing population of 170,000 inhabitants and is located 110 miles North East of Rio de Janeiro.
- It is the main operational base for exploration, drilling and extraction of more than 80% of all offshore oil in Brazil.
- It is also home to the Petrobras Campos Basin business unit and over 3,500 oil-related businesses.
- Due to helicopter traffic to and from the oil rigs, Macaé has the second-busiest airport in Brazil. after Rio and São Paulo.

Figure 1.4: The first four sentences of the article ranked fourth in the search result of the first query

Now, assume that another search query was made by replacing the word *petroleum* with *oil* rendering the new query "*Oil Extraction Brazil*". Since *oil Extraction* and *Petroleum Extraction* effectively refer to the same process, the results should still meet the user's needs. The top four results obtained for the new query from Google are illustrated in Figure 1.5.

Brazil Energy Data, Statistics and Analysis - **Oil**, Gas **...**
According to the **Oil** and Gas Journal (OGJ), **Brazil** had 12.6 billion barrels of **...** Most of**Brazil**'s crude **oil** production is offshore in very deep water and **...**
www.eia.doe.gov › International › Country Analysis Briefs - Cached

Brazil girds for massive offshore **oil extraction**

6 Dec 2009
Everything about the shipyard here is colossal -- the 4000-man workforce, the billions sunk into it in capital costs, the half-finished 10-story-high **...**
www.washingtonpost.com/wp-dyn/content/article/2009/12/06/AR2009120602442.html?wprss=rss_business- Related videos

Brazil-Arab News Agency - **Brazilian oil** and gas production broke **...**
27 Dec 2010 **...** São Paulo – **Brazilian oil** and natural gas production reached a record in November, according to figures disclosed this Monday (27th) by the **...**
www2.anba.com.br/noticia_petroleoegas.kmf?cod=11199044...0 - Cached

Brazil Oil **- production - Economy**
Facts and statistics about the **Oil** - production of **Brazil**. Updated as of 2010.
www.indexmundi.com › Brazil › Economy - Cached - Similar

Figure 1.5: The Top 4 search result links for the search query "Oil Extraction Brazil"

The first link now points to a webpage giving background about Brazil's oil production and oil reserves. The first four sentences are displayed in Figure 1.6. When comparing the top result of this query with the top result of the previous query, one can notice that more detailed information is presented in this article. The first sentence in Figure 1.2 mentions that the proved oil reserves in Brazil represent 0.96% of the world's reserves. Also, it can be noted that some information provided in both articles is repeated. For example, the essence of sentences 1 and 2 from the first query's top ranked summary is contained within sentence 1 of the second query's summary. In sentence 1 of the second query's summary, it is mentioned that Brazil produced 12.6 billion barrels of proven oil reserves in 2009. Also mentioned is that Brazil is the second-largest oil producer in South America after Venezuela. This merge of information into one sentence allows for summary 2 to display more information in the subsequent sentences than summary 1.

Brazil Energy Data, Statistics, and Analysis

- According to the Oil and Gas Journal (OGJ), Brazil had 12.6 billion barrels of proven oil reserves in 2009, second-largest in South America after Venezuela.
- The offshore Campos and Santos Basins, located on the country's southeast coast, contain the vast majority of Brazil's proven reserves.
- In 2008, Brazil produced 2.4 million barrels per day (bbl/d) of oil, of which 76 percent was crude oil.
- Brazil's oil production has risen steadily in recent years, with the country's oil production in 2008 about 150,000 bbl/d (6 percent) higher than 2007.

Figure 1.6: The sentences contained within the Brazil Energy Data, Statistics and Analysis webpage

The second link points to an article from the Washington Post titled "Brazil girds for massive offshore oil extraction". Figure 1.7 displays the first four sentences extracted from the article. It can be noted that more than half of the first sentence is not relevant to the main queried subject. The second sentence talks about the discovery of new offshore oil reserves and its expected effects on the country. Third and fourth sentences are about the difficulties faced with extracting oil from the offshore reserves and the efforts made to overcome these difficulties.

The third link in the search result points to the translation of a short article written by a Brazilian author. The first 4 extracted sentences are shown in Figure 1.8. In this summary, I note that different information is presented than what was presented in the first two articles. Daily and monthly statistics are also shown in this article.

The fourth link in the search result points to a webpage displaying charts and graphs about the average oil production per year in Brazil without much text in its contents.

Brazil grids for massive offshore oil extraction | The Washington Post

- Everything about the shipyard here is colossal -- the 4,000-man workforce, the billions sunk into it in capital costs, the half-finished 10-story-high production platforms.
- But then, so is the challenge facing Brazil's state-controlled energy company, Petrobras: developing a group of newly discovered deep-sea oil fields that energy analysts say will catapult this country into the ranks of the world's petro-powers.
- The oil pools are 200 miles out in the Atlantic and more than four miles down, under freezing seas, rock and a heavy cap of salt.
- Petrobras, which until recently was little known outside oil circles, has launched a five-year, $174 billion project to provide platforms, rigs, support vessels and drilling systems to develop tens of billions of barrels of oil.

Figure 1.7: The first four sentences contained within the Washington Post article

Brazilian Oil and Gas Production Broke Record in November

- Brazilian oil and natural gas production reached a record in November, according to figures disclosed this Monday (27th) by the National Petroleum, Natural Gas and Biofuel Agency (ANP).
- Daily oil extraction reached 2.089 million barrels, and daily gas extraction reached 66.2 million cubic metres.
- Oil production grew by 5.2% compared with November 2009 and by 4.6% compared with October this year, according to the ANP
- In the case of gas, there was growth of 12% over November last year, and 2% over October 2010.

Figure 1.8: The first four sentences extracted from the article ranked as third in the search result

A careful examination to the above summaries reveals the diversity of information presented in each and the differences and similarities among them. Although the main themes of most of the summaries do partially address the user’s two queries, it can be noted that the focus of the articles differ in many cases. This may be reasonable to expect since each article was originally written by a human author with a preset focus or goal meant to be primarily addressed in the article. Now, addressing the user’s specific query by browsing through the differently-focused articles with their authors’ diverse interests may be recognized as one of the challenges of Text Summarization. To properly identify the main theme of each article and extract the most useful parts to the reader according to his/her query, an understanding of the articles’ contents and the user’s query is needed.

A sentence may contain phrases of different interests to the user. An example for this is the sentence mentioned above in Figure 1.7 which carries important and relevant data in part, while in another part of the same sentence irrelevant information is presented. To save the user’s time and reduce the summary’s length, optimal summaries would present only the important and relevant parts of a sentence to the user. I present in this thesis how a Sentence Simplification Module (SSM) can be integrated with the summarization framework I propose to further help in condensing summaries and improving the overall performance of framework.

The example of using the different queries *Oil Extraction Brazil* and *Petroleum Extraction Brazil* emphasizes the need for a proper understanding of the query. When looking at the terms *Oil* and *Petroleum* individually, the definition of each may vary according to the background and culture of the reader, and the context they are placed in. By examining the definitions given in the Oxford Dictionary[1] for each, the meaning of Petroleum is *a liquid mixture of hydrocarbons extracted from earth which can be refined to produce petrol, diesel and other types of fuel*. The definition of the term Oil is *any viscous liquid which are insoluble in water*. From the given definitions, it can be noted that *Petroleum* refers to

[1] http://oxforddictionaries.com

crude oil extracted from earth while *oil* is a broader term that can cover engine, cooking, vegetable and other types of oils. However, in normal everyday usage, the two can refer to crude oil especially when coupled with certain terms such as “discovery” or “extraction”. In WordNet, one of the senses of the word *Oil* is *Petroleum*. In Wikipedia, “*Oil extraction*” and “*Petroleum extraction*” both refer to the article titled “*Extraction of Petroleum*”. In the two queries supplied to Google above, the results of the queries containing the two phrases differed in each case even though *Oil Extraction* and *Petroleum Extraction* both refer to the same process. In addition, it was found that the top four results for the former query were much more direct in answering the query than the latter.

A significant part of this thesis describes several new algorithms and methods used for better machine understanding of human text, or in other words handling the semantics of text documents and users’ queries. Just like in the previously mentioned example when replacing one query with another required background knowledge from the user to know that both are effectively referring to the same process, machines do need a way to learn or know that. For this, I show the use of different knowledge repositories in my algorithms and the implemented system that perform *Semantics Analysis*.

From the previously-mentioned example, it is evident that there are at least two generic problems at hand: (1) is finding the most relevant documents to the user’s needs, which is usually addressed by search engines, and (2) is extracting the most useful and important information from within the found documents. Researchers in the field of Text Summarization have focused on addressing the second problem in their work since at least the 1950s [1]. Starting with Luhn’s work [1] with IBM until very recently, as described in chapter 2, statistical measures have been commonly used without reliance on external knowledge for semantic analysis.

In this thesis, I describe a framework addressing the summarization problem with different and new approaches utilizing external knowledge to enhance the machines understanding of human languages and text. Semantic Analysis and Inferential Interpretation (words grouping according to the inferred meaning and context) are steps of the processes implemented in the framework to aid in this task. The methodologies implemented are also extended and applied to several related fields such as Documents Classification and Word-Sense Disambiguation.

The techniques implemented in my framework rely on the detection of direct and indirect semantic representation of terms, sentences and documents. This allows for the discovery of the main topics within a document and subsequently shifting the focus of the summarization according to the user's query, if provided, or the main theme of the document. A summary is thus generated by extracting the sentences containing the most important concepts and ignoring on the other hand the unimportant ones.

1.2 Aims and Contributions of the Thesis

The main questions being addressed by this thesis are: is it possible for summarization systems to effectively generate a summary that captures the essence of the documents it represents and reflects the user's specific goals as indicated in his/her query? To what extent can a system utilize external repositories to aid in Natural Language Processing-related applications, and specifically Automatic Documents Summarization? Is it possible to capture the semantic knowledge present in the external knowledge repositories and connect them to textual documents?

This thesis examines the use of different features extracted from within text documents and the enhancement process of these features by employing external repositories represented by WordNet and Wikipedia. The main contributions of this thesis are the following:

- The relationship between terms, and a composite of terms, is quantified and weighted through new algorithms allowing for grouping the terms, phrases and sentences based on the semantic meaning they carry.
- A framework is proposed to generate and extract features from text documents using external knowledge repositories such as WordNet and Wikipedia. This allows for embedding terms relations and human knowledge in the implemented system. Such features are not usually present in statistical-based systems and Bag-of-Words (BOW)[2] which do not rely on external knowledge repositories.
- The use of the framework in Automatic Text Documents Summarization allows for improvements of the performance and accuracy of the system when evaluated against many other systems.
- The use of Wikipedia in the process of generating features for text fragments allowed for the introduction of human knowledge *Concepts.* Quantifying the relationship between the detected concepts with the methods I propose allow for a better Automatic Semantic Interpretation of Text.
- The adaptation of several novel methods to enhance the diversity and reduce redundancy in the generated summaries. As illustrated in the evaluations performed, the results obtained suggest a better performance of the summarization system when compared against others that do not employ the implemented redundancy-diversity methods.

[2] Bag-of-Words (BOW) models treat text as an unordered collection of terms without taking into consideration their grammatical structure or their order of appearance within the documents they belong to.

- Extending the implemented summarization framework to apply the basic methodologies to other related fields such as Word-Sense Disambiguation and Documents Classification. Evaluations are also performed and the results suggest a competitive performance against other systems.
- The ability to detect main topics mentioned within free text and anticipating the relatedness of the detected topic to the main theme of the text and to what extent they are of interest to the reader.
- The introduction of a Sentences Simplification Module (SSM) and its integration into the Wikipedia-based summarization framework. Sentence splitting, compression and candidate selection are among the main tasks that were implemented into the summarizer.
- Participation in the Text Analysis Conference (TAC)[3] summarization task in two years: a WordNet-based system was used in 2008 while a Wikipedia-based system was used in 2010. The evaluation results obtained indicate competitive performance for both systems, with the Wikipedia-based having a higher rank than the WordNet-based system.

The following publications have been published while working on this thesis. They are mentioned here to provide a reference to the different sections covered in this thesis and the experiments that were conducted. They also give details about any ideas that may have been included briefly within this thesis.

[3] Text Analysis Conference (TAC) is a series of evaluations and workshops organized by National Institute of Standards and Technology (NIST) to encourage research in the field of Natural Language Processing by providing large test collections and common evaluation procedures. Summarization is among the tracks covered by TAC. It includes diagnostic and component evaluations situated within the context of end-user tasks. http://www.nist.gov/tac/

- *A. Bawakid, M. Oussalah*, Sentences Simplification for Automatic Summarization In *Proceedings of the 10th IEEE International Conference on Cybernetic Intelligent Systems 2011*, Sept 2011, London, UK

  This paper provides details on the implementation of the SSM module and rules I applied for simplifying sentences. It overlaps significantly with chapter 6.

- *A. Bawakid, M. Oussalah* . Summarizing with Wikipedia. *Proceedings of the Third Text Analysis Conference (TAC 2010)* Feb 2011, Gaithersburg, Maryland, USA

  This paper gives an overview of the Wikipedia-based summarizer I used to participate in TAC 2010 and the obtained results in comparison with the rest of the participants. A full description of the system, its implementation and the results gained is provided in Chapter 5.

- *A. Bawakid, M. Oussalah*, Using Features Extracted from Wikipedia for the Task of Word Sense Disambiguation In *Proceedings of the 9th IEEE International Conference on Cybernetic Intelligent Systems 2010*, Reading, UK

  This paper details how the Wikipedia-extracted features are used in the task of Word Sense Disambiguation (WSD) to disambiguate topics mentioned within text documents and how it was evaluated. The findings of this paper are expanded in Chapter 5 with more details on how the features weights were chosen and improved.

- *A. Bawakid, M. Oussalah* Centroid-based Classification Enhanced with Wikipedia In. *Proceedings of The Ninth International Conference on Machine Learning and Applications (ICMLA 2010)*, Fairfax, USA

This paper outlines the main evaluations that were performed to test the effectiveness of the features constructed from Wikipedia. A description of the relevant parts of the system and evaluations are provided in Chapter 7.

- *A. Bawakid, M. Oussalah* A Semantic-Based Classification System. *Proceedings of the 8th IEEE International Conference on Cybernetic Intelligent Systems 2009*, Birmingham, UK

  This paper describes the WordNet-based similarity measures given in Chapter 4 and applies them to the application of documents classification.

- *A. Bawakid, M. Oussalah* A Semantic Summarization System: University of Birmingham at TAC 2008. *Proceedings of the First Text Analysis Conference (TAC 2008)*, Gaithersburg, Maryland, USA

  This paper details a relevant study in which linguistic qualifiers were used with the WordNet-based summarizer to summarize systems. Some aspects of the implemented system are detailed in Chapter 4.

1.3 Structure of the Document

The structure of the thesis can be segmented into five different parts. The first part for the introduction, overviews related research work and describes features construction. The second and third parts contain the thesis main contributions regarding how external repositories, namely WordNet and Wikipedia, can help in the task of summarization. The fourth part describes several related applications that utilize the developed methodologies and can either help with summarization or test the usability of the extracted features. The fifth part draws the conclusions of this thesis. These five parts have been structured into chapters as follows:

- **Part I: Introduction, Background and Context**
    - *Chapter1: Introduction*
    - *Chapter 2: Background and Related Work.* This provides background about related research work and state of the art in Summarization and summarization evaluations.
    - *Chapter 3: Features Generation and Selection*. This chapter describes the need for features generation and selection when using external ontologies for automatic summarization. An overview about the stages involved and how semantic distance is derived from WordNet and Wikipedia are also given.
- **Part II: Using WordNet for Summarization**
    - *Chapter 4: Summarization Aided with WordNet.* This chapter describes several metrics for computing the similarity between sentences with the aid of WordNet. The implementation of the built summarizer, evaluations performed and improvements added via redundancy checking and diversity enhancement are also given.
- **Part III: Using Wikipedia for Summarization**
    - *Chapter 5: Summarization Aided with Wikipedia.* This chapter provides an overview on how features are extracted and built for Wikipedia for use in different applications. The extracted features are used in the built summarizer and the evaluation results of its performance are also reported.
    - *Chapter 6: Sentence Simplification for Automatic Summarization.* This chapter extends the previous chapter by introducing SSM to further condense the summaries and allow for inclusion of more information.

- *Chapter 8: Using SSM for Summarization.* This illustrates the effects of applying SSM to several sample sentences and its usability in the application of summarization.

❖ **Part IV: Related Applications**

- *Chapter 5.* The built features from Wikipedia were used in the task of WSD for two reasons: to get a better view of the effectiveness of the extracted features and to also aid in the task of automatic summarization. In many cases when analyzing documents for summarization, ambiguous words are encountered and a module that effectively handles these types of terms would positively affect the overall performance of the summarizer.
- *Chapter 7: Classification Aided with Wikipedia.* To test the effectiveness of the built features from Wikipedia, text classification was used as the first application to explore. I used the constructed features to build a classifier and evaluated its performance.

❖ **Part V: Conclusions and Future Work**

- *Chapter 9: Conclusions.* This chapter draws the conclusions for this thesis and potential future work.

The relationships among the parts and the chapters are displayed in Figure 1.9. These relationships and links serve the purpose of highlighting the flow of reading required. For example, if a person is interested in learning only about how WordNet was used for summarization in this thesis, then Parts I, II and IV need to be read in that order.

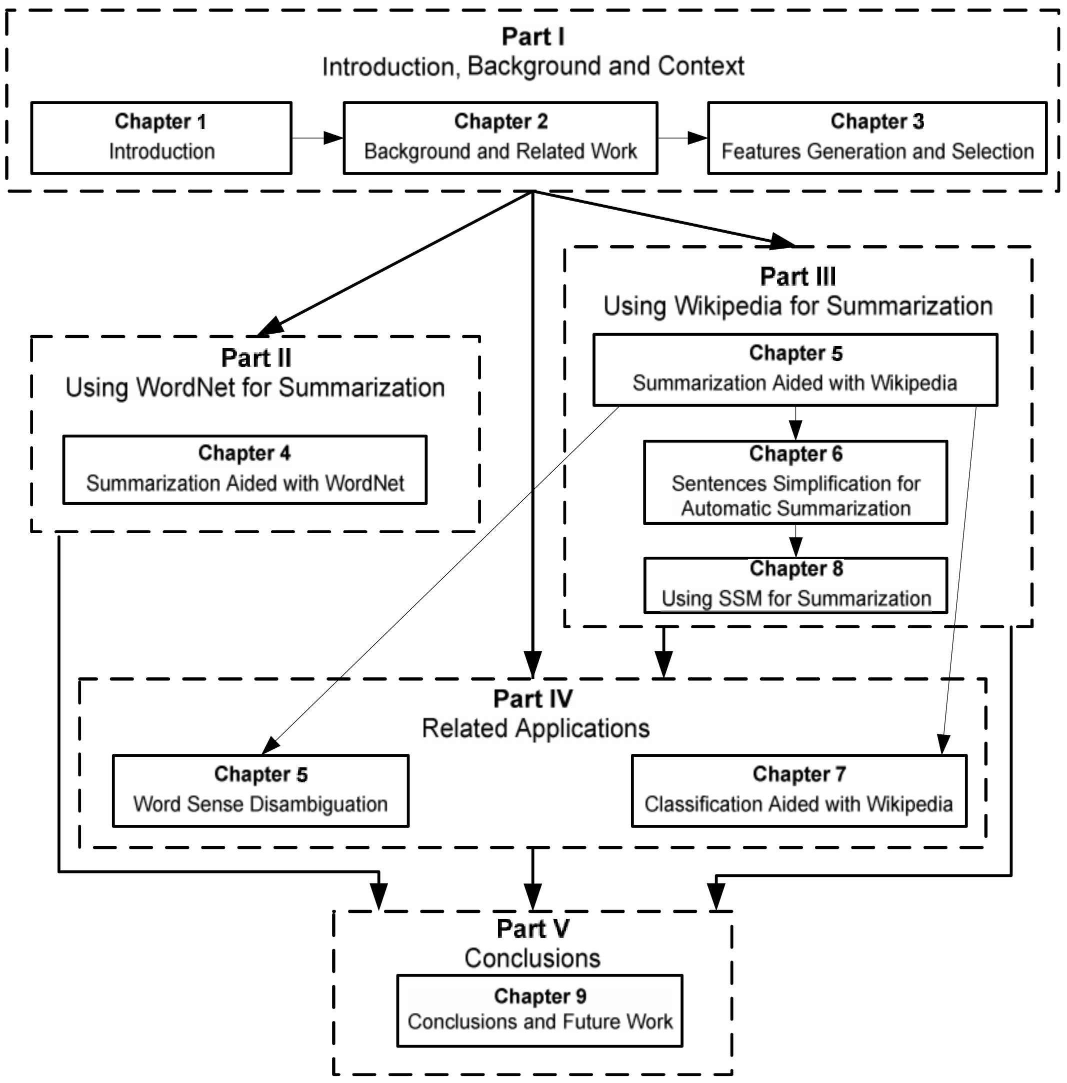

Part I
Introduction, Background and Context
Chapter 1
Introduction
Chapter 2
Background and Related Work
Chapter 3
Features Generation and Selection
Part II
Using WordNet for Summarization
Chapter 4
Summarization Aided with WordNet
Part III
Using Wikipedia for Summarization
Chapter 5
Summarization Aided with Wikipedia
Chapter 6
Sentences Simplification for Automatic Summarization
Chapter 8
Using SSM for Summarization
Part IV
Related Applications
Chapter 5
Word Sense Disambiguation
Chapter 7
Classification Aided with Wikipedia
Part V
Conclusions
Chapter 9
Conclusions and Future Work

Figure 1.9: Relationships among thesis main parts and chapters

In this thesis, I chose WordNet and Wikipedia as the main ontologies due to the abundance of human concepts available in both. The human knowledge that exists in both ontologies is made available to machines through the framework and algorithms I describe in this thesis. The superior inferring capability of humans is counter measured by introducing the abundant human knowledge to machines all at the same time through the proposed algorithms and methods.

I use WordNet, a hierarchically-structured repository that was created by linguistic experts and is rich in its explicitly defined lexical relations. With WordNet, algorithms for computing the semantic similarity between terms are proposed and implemented. The relationship between terms, and a composite of terms, is quantified and weighted through new algorithms allowing for grouping the terms, phrases and sentences based on the semantic meaning they carry. These algorithms are especially useful when applied to the application of Automatic Documents Summarization as shown with the obtained evaluation results. Several novel methods are also adapted to enhance the diversity and reduce redundancy in the generated summaries.

I also use Wikipedia, the largest encyclopaedia to date. Because of its openness and structure, three problems had to be handled: Extracting knowledge and features from Wikipedia, enriching the representation of text documents with the extracted features, and using them in the application of Automatic Summarization. First, I show how the structure and content of Wikipedia can be used to build vectors representing human concepts. Second, I illustrate how these vectors can be mapped to text documents and how the semantic relatedness between text fragments is computed. Third, I describe a summarizer I built which utilizes the extracted features from Wikipedia and present its performance.

I apply the methodologies proposed in this thesis to the application of automatic documents summarization. To evaluate the effectiveness of the different variations of the implemented summarizer, I participated in the TAC 2008 and TAC 2010 summarization tasks with runs from the WordNet-based and the Wikipedia-based summarizers. I report in this thesis the results of the evaluations performed and compare them against several baselines and the results of the other TAC participants.

Chapter 2
Background and Related Work

This chapter presents an overview on Text Summarization and the stages it involves in general. It then reports the main Automatic Summarization Systems which have been developed so far and outlines the major techniques being used. Afterwards, the major summarization evaluation methods are described. The challenges being faced by Automatic Text Summarization are then introduced and those that are addressed by this thesis are highlighted.

2.1 Text Summarization

For a person to manually generate a summary, s/he has to ***read and understand*** the original document first. Based on the understood events, facts or situations within the document, the ***important aspects*** are specified to meet the purpose of the summary. The summary would not contain all of the information present within the original document, but only those deemed to be important. This is obvious since the goal of the summary is to reduce the amount of information present in the original documents. After specifying the important aspects within a document, the summary is then produced in a ***suitable output format***.

The generic stages of summarization mentioned above have been addressed in the previous work of Luhn [2] in which the author claims that summarization involves three aspects in general: ***input***, ***analysis*** and ***output***. For the input aspect, humans usually require an understanding of the natural language the text document is written in. Analysis would require determining the purpose of the summary and the target audience.

Synthesizing a suitable output form for the summary would then be the last step before it is presented to the user.

For each of the mentioned summarization aspects, there are many factors to consider regardless of whether the summarizer is a human or a machine. For the ***input*** aspect: Depending on how the document is structured, the summarizer would have to decide how to approach reading the document. For instance, the headers of chapters or labels of Figures and Tables may contain information which is useful in the analysis stage. Metadata of some documents such as the keywords of HTML webpages may be beneficial, too. If the document was classified and the class or domain it belongs to is accessible by the summarizer, it may be possible to utilize knowledge restricted to that domain to aid in the analysis and output stages. The language of the text documents may also have an effect on the summarizer. Human summarizers would usually require an understanding of the language the document was written in. Machines on the other hand, lack the full and deep natural language understanding which humans normally possess. In addition, human summarizers usually have background information and common sense knowledge about the world and possibly the subjects in the document allowing them to infer between sentences. Take the following two sentences as an example: “Adam ordered a delivery pizza. He liked its taste very much”. It can be inferred that the pizza was prepared, cooked, delivered to Adam and then that Adam ate the pizza. This inferring capability requires common sense knowledge and it is something that machines lack as was noted by Lenat in [3].

In the ***analysis*** stage, the summarizer would evaluate and select the important parts from a document. The purpose of the summary would have an effect on how it is analyzed. For example, user specific summaries would normally be tailored to match the user’s interests, profile or background. Generic background summaries assume a poor user background about the subject and hence an overview of the subject or the content of the document is included in the summary. An update summary for a news event on the other hand, would include only new key updates with the assumption that the user has read previous older articles. The user goal may also affect how the document is analyzed. When a user searches for specific information, the summarizer would focus the summary to mostly related information to what the user has searched for. The number of documents to be summarized is another aspect affecting how analysis is performed. With single-document summaries, the structure of the document may give greater impact on the final generated summary than of the multi-document summary. This is especially evident if the structures of the different documents vary greatly.

The shape and ***output*** of the summary can take different forms. The summary can be in the form of extracts containing unaltered pieces from the original text such as full sentences or paragraphs. It can also be in the form of abstracts, where new phrases or sentences are created. The length of the summaries also varies based on the intended purpose and the compression rate desired. The shape of the summary can be in the form of complete sentences, or simply phrases as in news headlines. Summaries can be presented in the form of simply Text, or Text and other contextual information such as hyperlinks or related sentences depending on the user interface being used.

In the work performed for this thesis, the focus is mainly on generating extract-based single-document or multi-document summaries. The extracts are in the form of sentences taken from the original text. The use of the Sentences Simplification Module (SSM) allows me to adapt the system and generate abstractive summaries as described in chapter six. The summaries generated are domain-independent. Due to the repositories used being in English, the developed framework is currently restricted to only the English language.

No prior knowledge is assumed about the users and the summaries are only influenced by the text documents contents and the user query if supplied.

2.2 Automatic Documents Summarization Systems

As described in section 2.1, three stages are being shared between all summarizers regardless of whether they are machines or humans: inputting documents and task-specific data, analysis and outputting the summary. With Automatic Documents Summarization systems, it is possible to rephrase these steps and draw them as illustrated in Figure 2.1. The shown steps are shared and adapted by all of the available summarization systems. The details of the implementation of these steps are what make a system different from another. The figure illustrates that there are three main components: ***Parser***, ***Analyzer*** and ***Synthesizer***. The first stage may sometimes be referred to as the Preprocessing stage while the later is often called post processing.

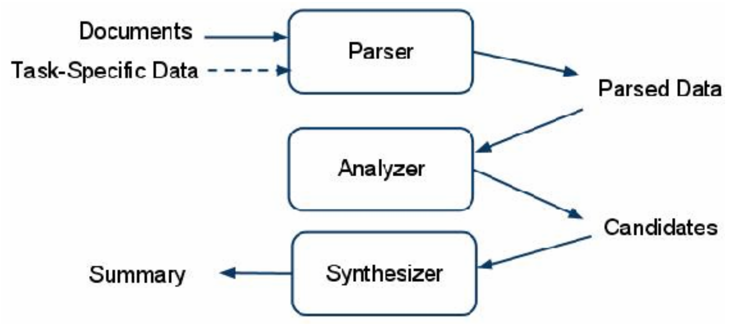


Figure 2.1: Summarization Systems Architecture

The ***parser*** is first fed with the documents to be summarized. Task-specific data such as user's queries and compression rate may also be fed into the system. The parser then parses the fed data and prepares the documents in a suitable format acceptable by the analyzer. If there are unnecessary data included in the fed documents which are not handled by the system, they are removed at this stage. Usually, the parser applies a sentence boundary detection module to the fed documents to segment the sentences. However, in some Bag-of-Words (BOW) systems the sentence boundary detection module is not needed. After segmenting the sentences, additional information may be tagged to each sentence such as its order of appearance within the document and/or the paragraph if several exist. If the system requires additional data as part of its input in the analysis stage, tagging is usually performed within the parsing module. After tokenizing the terms within each sentence, tags such as the terms Part-of-Speech Named Entities (Locations, Organizations and Persons) may be applied.

The generated data are then fed to the ***analyzer*** where the core algorithms of the system are applied. A weight is usually attributed to each of the features detected or generated for each sentence. A score is then assigned to each sentence representing its importance. A sentence score is usually the sum of the weighted features scores. For some systems which produce abstractive summaries, sentences simplification, splitting, trimming or compression may be applied in this stage, too. In cases where the system is using external corpora or knowledge repositories to aid in summarization, external corpora are usually employed in this stage. Scored candidates of clauses, phrases or sentences are the result of the analyzer.

The role of the ***synthesizer*** is to organize the scored candidates and present them in a form suitable to the user's needs. If a compression rate or words/sentences limit is specified, the synthesizer ensures that the output meets the given conditions. For single-document summaries, producing the summary is usually straightforward and is accomplished by choosing the highest ranked sentences according to their scores. For multi-document summaries, the process is usually more complex as it involves checking for redundancy, diversity and relevance to the user's specific needs.

2.3 Related Work

A fairly large number of automatic summarization systems can be found in the literature. Many of these systems address the summarization problem depending on the desired type of summaries. In general, the systems can be classified into two categories: Extractive and Abstractive summarizer. Extractive-based methods focus more on producing the best *content* for the summary. Abstractive methods emphasize the need for having *restructured text* and usually require an advanced level of language-dependent operations. It is also possible to classify the different summarization approaches into other different categories based on the specific characteristics of each approach as outlined in [4], [5], [6] or [7]. I provide here a sketch of the major summarization approaches that pioneered the field and the systems that employed these methods using a classification similar to the one presented in [7]. Summarization methods can be classified as those using mainly surface level features, machine learning approaches, natural language analysis methods, abstraction, topic driven, graph-based, and LSA-based methods. I also include a subsection for other methods which adopt a combination of these methods depending on the details of summarization task at hand. I follow by presenting the limitations of the approaches most related to my work and highlight the main differences between what I apply in this thesis and what was previously presented.

2.3.1 Surface-level Features

Approaches that tend to rely on surface level features extracted from the test documents are among the first that were applied in the text summarization field and date back to the 1950s with Luhn's work [1] at IBM premiering them all. These approaches are generally extractive and utilize features such as the position of sentences, words frequencies, the presence of cue words and overlap with document title or query. In the following subsections, I describe these features and their use in the early work of the summarization field.

2.3.1.1 Word Frequency

The assumption made by approaches relying on words frequencies is that important parts of text would likely contain words occurring with high frequency. The first summarization approach proposed by Luhn in [1] at IBM implements this method. The algorithm he devised relies on the idea that frequently occurring terms reflect the main theme of the document. Commonly used terms in the English language, which are previously prepared in what is called Stop Words List, are removed from text. Sentences are assigned scores reflecting the frequency of the terms they contain. After ranking all sentences based on their scores, a summary is formed by selecting the top ranking sentences. In the work of [8], a threshold was introduced giving words with frequencies above the threshold a higher weight than the rest. In [9], the assigned term frequency weights were affected by their positions within the document. In [10] , they used terms frequencies to define what they called *Thematic Words*. A small number of thematic words are selected and each sentence score is affected by the number of thematic words it contains. In [11], groups of *Topic Signatures* were used where each topic signature consists of a topic and related terms. For example, the topic “restaurant visit” has the related words “table”, “menu”, “order”, etc. When analyzing a document, words are replaced with their topics based on the topic signature groups. These groups are built by considering the frequencies of the different terms within the previously classified documents.

Later on in the 1980s, it was observed by Salton [12] that common terms within a domain, or within a collection of documents to be evaluated, may affect the performance of the terms frequencies method proposed in Luhn’s work. It was suggested in an added feature that the relevancy of a term to a document is inversely proportional to the number of documents in the corpus containing that term. This is when the *Inverse Document Frequency* (IDF) measure was introduced to the field. And so, the Term Frequency (TF) formula was revised to include IDF by changing it to TF x IDF. Thus, a sentence score is computed by summing the scores of its individual terms and a summary would then be generated by simply aggregating the highest-scoring sentences.

2.3.1.2 Position

Another feature exploited by many summarization systems is the location of sentences within a document. The structure of the document is usually taken into account while scoring sentences. For some methods, sentences appearing at the beginning of a document are assumed to be more important than the rest. This goes along with the results of the study described in [8] in which 200 paragraph were examined. It was found that the topic sentence came as the first one in 85% of the examined paragraphs and as the last sentence in 7%. Also in [13], it was found that topic sentences tend to appear very early or late in the documents. Variants of this feature were used in several other summarization systems including [9], [14] and [15].

2.3.1.3 Cue words and phrases

The presence of certain words or phrases in a sentence may indicate the importance or irrelevance of a sentence. In the work of [13], bonus and stigma words were defined. The existence of bonus words such as “significant” or “important” in a sentence indicates its importance. Stigma words such as “hardly” or “impossible” indicate the irrelevance of a sentence giving it a higher chance for being excluded from the summary. The work of [16] proposed a system called ADAM for excluding sentences from summaries based on the existence of certain phrases. In [15], lists of cue phrases have been built manually to indicate the significance of a sentence. Then in [17], the process of building lists of cue words and phrases was automated in a trainable features combiner that learns from summary examples. Other systems which used this feature include [18], [19] and [20].

2.3.1.4 Overlap with title or query

With this method, words or clauses appearing in the document title or user query are determined. Sentences having words in common with the title or user query, if provided, are given higher weight than others. Systems that employ this feature include among others [21] and [11].

2.3.2 Machine Learning Approaches

In many extractive summarization systems, several algorithms have been adopted which employ machine learning and statistical techniques. Some employ Naïve-Bayes methods which assume the independence between the features used while others do not make the same assumption. Some use other techniques such as Decision Trees, Hidden Markov Models, Log-Linear models and Neural Networks. I give an overview on these techniques next.

2.3.2.1 Naïve-Bayes Methods

A Bayesian classifier was employed in [10] to compute the probability of whether a sentence in the source document should be included in the summary. The classifier was trained with technical documents which were prepared in the form of 188 document-summary pairs. Every sentence was assigned a score based on the probability computed with a set of features that include the structure of phrases within a sentence, sentence length, position and presence of uppercase and cue words. Only the top ranked sentences are chosen to form the summary. In the results of their analysis, it was found that the use of only the sentence position, cue words and sentence length features produced the best results. In [22], another Bayesian summarization system was developed which applied several syntactic-based rules to trim chosen sentences and reduce their length. It was found in several studies such as [23] and [24] that the use of purely syntactical features in Bayesian classifiers may slightly improve the precision in the low recall end, but not in the high recall end.

2.3.2.2 Decision Trees

In a step moving away from the assumption that features are independent from each other, summarization systems have emerged that use decision trees instead of Bayesian classifiers. With decision trees, tree like models of decisions or graphs are created as support tools. In [25], a decision trees-based system extracted sentences from source documents which were matched against reference human summaries for evaluation. The system introduced new features affecting the score computed for each sentence. These features include the existence of numerical data, weekday or month, pronoun or adjective, overlap with a user query or corpus most salient terms. The author performed evaluations against several baselines that include the combination of some of the mentioned features. The overall results he obtained indicate the prevalence of the used decision-tree method. Other variations of the tree-based method were also used in [26] and [27].

2.3.2.3 Hidden Markov Models

To capture the dependencies between sentences within a document, several systems have explored the use of Hidden Markov Models (HMM) creating a sequential feature-based system with unobserved or hidden states each having a probability distribution over all possible outputs. The OnTopic system described in [28] uses HMM for assigning topics to document and classifying broadcast news. In [21], a summarization system was described which employs three features: sentence position, sentence length and the likelihood of having the sentence terms in the summary given the source document terms. Training corpus was required for that system with pairs of original document-summaries clearly labelled. Also in [29], [30] and [31] other variations of HMM-based summarizers were used. In [32], HMM-Hedge was proposed which uses distinct language models of news stories and headlines but does not requires manual pairing of stories and summaries in the provided corpus. Other syntactic systems such as CLASSY [33] rely on the statistical Hidden Markov Model (HMM) in sentences selection. Also, it uses a simple rule-based sentences compression module by removing non-important phrases from sentences based on the existence of cue-words.

2.3.2.4 Log-Linear Models

The use of log-linear models in summarizers also emerged as a deviation from the assumption that features are independent. It usually works by transforming a system of equations through finding logs of all variables and approximating the results with linear equations. It has been used in the summarizer of [34] where the authors claimed that the results they obtained outperformed those obtained with Naïve-Bayes methods. In [35], a log-linear model was used in an iterative process to automatically select the input features that increase the predictive capability of their system. The results they obtained showed a remarkable increase in the performance of their system when compared against other similar summarizers which do not utilize the log-linear models. In [36], log-linear modelling was used to aid in generating sentiment summaries.

2.3.2.5 Neural Networks

Neural networks are non-linear statistical data models used to model complex relationships between inputs and outputs and detect patterns within text. The structure of the networks usually changes based on the information processed during the training stage. In DUC 2007, a summarizer called NetSum [37] using neural networks-based algorithm has demonstrated a performance significantly exceeding those of the provided baselines. The algorithm used a pair-based sentences ranker called RankNet [38] for ranking all sentences in the source documents based on their importance. Training is performed using a modified version of the back propagation algorithm [39] which is based on the gradient descend method described in [37]. Other summarization systems that also employ neural networks in their algorithms include [40] and [41].

2.3.3 Natural Language Analysis Methods

I describe here a set of methods that involve complex language analysis without the need for machines training and learning. The analysis performed can be classified as entity-level analysis and discourse level analysis. Entity level analysis is usually performed by building an internal representation of text through the use of text entities and modelling the relationships between them. Discourse level analysis on the other hand models the overall structure of the text and its relation to the goal of the summary [42]. I give an overview next on each of the mentioned analysis types.

2.3.3.1 Entity Level

With entity level analysis, text is represented with entities and the relationships between these entities are identified. Entities can be in different forms including simple terms, n-gram words and Named Entities (NE). The relationships between the entities can be in the form of similarity, co occurrence relatedness, logical relations or co reference relatedness [42]. In [43], the semantic similarity between the title or query and the rest of the documents is computed to score sentences and form the summary. Similarity between entities was initially performed through the detection of words overlaps as was implemented in [11]. The words overlap method applies its analysis on the explicitly mentioned terms within the original documents. If a document contains the terms “household”, “house” and “home”, each word would be treated separately from the rest. In [11], the authors attempted to address this problem by using WordNet [44] and the synonyms it contains. A similar approach was made by [45] and [46] where words were regarded important if they are deemed related to most words in a document with the help of WordNet. WordNet based semantic similarity measures were introduced in [47] and [48] and used in several other summarizers as in [49], [35] and [50].

In the work of [51], external thesauruses were employed to form lists of synonyms and hyponyms and use these lists to find the relationships between entities. Entities can be formed with n-grams spanning short or long distances of text. The relationships between these entities can be any of the mentioned forms or a combination of them as illustrated in [52]. In [46], important entities are detected through the analysis of syntactic and grammatical structure of sentences.

Words co occurrence is another method used for defining relationships between entities. In [53], it was implemented by examining the similarity between the contexts of the target entities. The larger the number of overlapping terms between the compared contexts, the more similar the target entities are. Co reference relations between entities were exploited to quantify the relationships between the phrases or sentences containing the referred entities. This was implemented in the query-based summarizer described in [54].

The use of logical relations as another form for detecting the relationship between entities has been explored in the literature in [42]. Agreements, contradictions and entailments are examples of logical relations. A system built by [55] uses logical relations to summarize documents. Textual Entailment (TE) is another form of relationships that can be defined between entities. It is said that text T entails H if the truth of H can be inferred from T. In [56], the use of TE was proposed and implemented in an algorithm for text summarization.

2.3.3.2 Discourse Level

Discourse-level analysis methods focus on studying naturally occurred language use and not custom made examples. They examine the relationships between texts fragments beyond the sentences boundaries and have two distinctive properties in common: cohesion and coherence. Cohesion accounts for the relationships between text fragments while coherence is represented in the form of relations between text fragments such as elaborations, causes and explanations as explained in more details in [57]. The first reported discourse level approach modelled text relations based on story grammars in [58]. In [59], a computational model of discourse for Japanese writings was proposed for extracting the discourse rhetorical structure and forming binary trees to represent the relations between texts.

The Rhetorical Structure Theory [60] was then used in subsequent summarization systems. In [61], the author devised a heuristic-based summarizer employing the RST theory and using basic features similar to what have been used previously for summarization in the literature. The author formed Rhetorical Structure Trees which represent relations between two non-overlapping text fragments: Nucleus and Satellite. The main difference between the two is that the Nucleus expresses what is more essential to the text's writer and that the Nucleus of a rhetorical relation is comprehensively independent of the Satellite but not vice versa [62]. The extractive summaries produced by that system were meant to be coherent and contain only salient information.

In the summarizer of [63], discourse markers were used to integrate aspects of coherence and cohesion in the same summarizer adding argumentative structure information to lexical chains. The performance achieved by that summarizer when compared with other baselines was not encouraging. In [64], this work was expanded by a hybrid summarizer that combines statistics criteria with linguistics and produces better results.

2.3.4 Abstraction

Producing extracts for summaries is the most common approach for summarization due to its simplicity when compared with abstraction. Abstraction differs mainly from extractive approaches by providing summaries having some degree of inference about background knowledge not necessarily present in the original document and presented in a different structure and format. Thus, abstractive summaries are usually shorter and more condense than extractive summaries as they are not restricted to the information present in the original source documents to reproduce summaries from. It has been reported in [42] that building an open-domain abstractive summarizer would require a very large knowledge base that would make it impractical to apply automatic abstraction on a large scale. This led to having most of the built abstractive summarizer being applied in a small scale to domain specific applications.

The first recorded automatic summarizer was extractive [1] and it was not until the SUMMONS system [65] emerged that an automatic abstractive summarizer came to existence [7]. The summarizer tackles a specific type of documents in a specific domain, namely news articles about terrorism. It works by using a previously built database with a template-based message understanding system, and does not directly handle raw text. The template used was manually prepared for that specific domain. The summarizer produces briefings by using connective phrases to merge relevant information about each event.

In [46], an analytical programme [66] was used to study a corpus of simplified sentences written by human professionals. The idea was to analyze the differences between summarized phrases and their long complete versions and compute the likelihood of having a syntactic part of a sentence being removed without affecting the meaning and grammatical structure of a sentence negatively. Other systems which relied more heavily on sentences compressions include [67] in which a rule-based compression system was introduced to help blind-readers browse quickly through text with the aid of a text-to-speech programme.

The work of [68] discusses different methods for abstractions. One of these methods is Pretty Printing which uses texts from templates to produces portions of summaries and makes them more user friendly. Graphical output was another abstraction method which provides users with outputs in the form of graphs and charts.

A discussion for how Natural Language Generation (NLG) can be performed and how it serves to produce summaries is provided in [69]. The STREAK and PLANDOC [70] summarizers have employed NLG for creating summaries for baseball games and telephone network planning activity. In [71], an abstractive summarizer was introduced by applying compression rules and a Maximum Entropy (ME) [72] classifier to generate meeting summaries. The results they obtained suggest that compressing sentences can lead to improvements in ROUGE scores. However, the best performance they obtained with their system is still quite low. Another study in [73] compared an extractive summarizer with an NLG-based abstractive summarizer by applying controversial measures and using controversial corpora. The results they obtained indicate that extractive summarization can give better results if the controversiality of opinions in a corpus is low. They suggest a mix of abstraction and extraction in summaries when the controversiality is high.

2.3.5 Topic-driven Summarization

Documents are usually prepared and written in such a way to cover individual topics or subtopics in separate sections. The documents are broken up either explicitly or implicitly into segments where each segment focuses on a separate theme or topic. Thus, it is intuitive to think that optimal summaries would cover as much of the topics that are of interest to the user as possible. The Maximal Marginal Relevance (MMR) metric proposed in [74] aimed to employ this feature. It works by clustering documents passages based on their topics and chooses passages of large coverage to extract sentences from while keeping redundancy low. It proposes to reward relevant sentences while penalizing redundant ones through the use of a linear combination of two similarity measures. MMR-based summarizers such as [75] account for a number of criteria for sentence selection such as content words, chronological order and similarity with document topic or query. They use BOW models such as Vector Space Model (VSM) or Latent Semantic Analysis (LSA) to represent documents and compute similarity between the source documents sentences and the query or documents titles using the cosine similarity measure. The sentences chosen to be included in the summary are usually highly similar to the query or document main topics. When a sentence is added to the summary, the redundancy is usually implicitly checked to ensure it is kept to a minimum level in the summary. The length of the generated summary can be controlled with a user supplied threshold.

In the work of [76], a modified version of MMR is used for multi document update summarization relying on a double maximization criterion. A rule based linguistic post processing stage is also applied to reduce the length of sentences and ensure cohesion. A study [77] which applied an MMR-based summarizer on Portuguese documents and compared it with an LSA-based summarizer found after evaluating both systems that

MMR produce better summaries than LSA when compared with human made summaries. Another modified version of MMR called Video-MMR was proposed in another study [78] in which the author extended the classical MMR algorithm for text summarization to videos by forming video summaries through rewarding relevant keyframes and penalizing redundant keyframes.

2.3.6 Graph-based Theories

With graph-based theories, text entities and the relations between them are represented in the form of undirected graphs. Each entity is viewed as a node in the graph. An edge between two nodes is drawn if a relation exists between these two nodes. The relation can be a cosine similarity above a threshold, relatedness, sharing a common term or any other type of relationships according to the implemented algorithm. In addition, it is possible to have different types of edges if more than one relation is used to connect edges. After drawing a graph, it is possible to view the sub graphs of connected nodes as clusters of topics. A generic summary would then be formed by selecting pertinent sentences from each sub graph for best coverage. Another aspect of the formed graphs is the number of edges for each node. The larger the number of edges for a node, the more central the node is and hence the more important is its associated text. When having more than one document represented in the graph, inter-document and inner documents relations may be viewable, too. The TextRank algorithm presented in [79] provides an example implementation for a graph-based technique in the field of Natural Language Processing. PageRank [80] is another example of a graph-based algorithm. The work of [81] and [82] applied graph-based techniques in their summarization systems.

2.3.7 LSA Methods

Latent Semantic Analysis (LSA) is a technique for capturing hidden relationships between documents and text fragments on the basis of their contextual use. It was introduced in the early 1990s in the work of [83] and has since been used in many NLP applications including documents clustering [84] and classification [85]. It was also used in several summarization systems as in [86] and [87]. In the LSA-based system described in [88], the interrelationship among terms and a set of documents are analyzed and captured to aid in forming summaries and compressing sentences. Anaphora resolution is also employed in another LSA-based summarizer [87] to revise the incorrect references before compressing the sentences.

Originally, LSA was introduced as a measure to overcome the words synonymy problem in documents and queries representations. It was found to be useful at reducing the problem of words synonymy through a better representation of descriptive features. However, due to not capturing discriminative features [89], it may not be best suitable for handling diversity and redundancy in summarization tasks especially for short documents.

2.3.8 Task-specific Approaches

In addition to the above mentioned summarization techniques, it is possible to classify summarization into other classes from other perspectives according to the generated output type (complete sentences, headlines, videos, graphs, etc.), targeted language (one language, cross-lingual or multi-lingual), documents types (text, speech, video, etc.), number of documents (single vs. multi document summarization) or whether summaries are for specific application with specific supplied parameters (query, thresholds, etc.). These other types of summarization methods may involve some of the methods described above or combine them with others. Below is an overview on different aspects of some of these types of summarizers.

Headlines: In [90], a system was built to generate headlines by applying a hybrid model merging a unigram language model for scoring sentences with a Bayesian classifier on features such as cue words and length. It parses all the sentences in the sources documents and aims to extract the most salient Noun Phrase (NP) using the hybrid model as the representative headline. In [91], the summarizer constructed summaries in a form similar to the news headlines by aggregating and trimming sentences containing topic terms. A similar approach was implemented in [24] and then extended in [92] by combining topic terms with sentences compressions. In [93], extractives summaries are formed through the detection of key topic terms within the text phrases.

Question Answering (QA): Some summarizers have been designed for tasks requiring answers to specific given questions. In most cases, query-based summarizers can be adapted to handle QA, too. In [94], a summarizer is proposed which applies a multi-stage question decomposition into simpler ones as a first step. The decomposition involves syntactic and semantic interpretations of the question which are passed to a summarizer to produce a summary for each simplified form of the question. Finally, a textual entailment system is applied to rank the summaries and choose the best as a representative for the complex question.

Graphs/Video/Speech: Many types of data can be modelled as graphs such as social networks and network traffics with attributes associated to each node and relationships defined for linking nodes. Most of the existing graph summarization systems use simple statistical methods for interpreting and summarizing graphs as illustrated in [95] and [96]. Some other summarizers attempt to detect frequent patterns to interpret the graphs and eventually produce summaries as in [97]. Other summarizers have also been proposed to summarize video as in the work of [78] in which a custom MMR algorithm is applied. For speech summarization, systems relied on Automatic Speech Recognition techniques for converting speech to text and then applying any text summarization approach desired. Speech summarization is naturally a more difficult task than text summarization since the text-speech conversion is almost always prone to errors. In addition, the conversion may not always include custom language markers such as commas, punctuations and questions marks making the preprocessing stage in most summarization system even more difficult. A study and analysis on these issues has been covered in [98], [99] and [100].

Language Dependency: Some summarization systems are designed to work with only one language and have the input documents and the summaries generated in the same language. These types of summarizers are common in the literature. Other summarizers are multilingual. They cover more than one language but the summaries generated are always of the same language as the input documents. Statistical and Machine Learning approaches are mostly followed for multi-lingual summarization. The SUMMARIST system [11] is an example of a multi lingual summarizer. Some other systems are cross lingual accepting input documents of language and producing summaries in another. These types of summarizers usually involve aspects from the Machine Translation domain. An example for an implementation of a cross lingual system is illustrated in [101].

Ill-formed Input: Informal spoken and written texts are not always similar in style and format to formal texts used and processed by many summarization systems. Documents such as emails, mobile phone text messages and transcribed speech are not always complete or grammatically correct. Some summarization systems have attempted to focus on this problem by transforming ill-formed texts into a more concise and formal format. Information retrieval metrics and syntax checking and transformation techniques are among the approaches used for tackling this problem. An example for a summarization system that makes assumptions to deal with ill-formed input is presented in [100].

2.4 Examples of Automatic Summarizers

I present here some of the most common commercial and academic summarization systems and give a brief description about each. I also highlight the targeted application for each of these systems and the languages they can handle. Table 2.1 presents a summary of the main public automatic summarizers and highlights their features.

MEAD	Multi document multi lingual	Downloadable from http://www.clsp.jhu.edu/ws2001/groups/asmd/
NewsInEssence	Online news summarization version of MEAD – English only	Online version http://www.newsinessence.com/
Newsblaster	Multi source multi lingual	Online version http://newsblaster.cs.columbia.edu/
Condensr	Multi source – English Only- Restaurants Reviews Summarization and Sentiment Analysis	Online Version http://www.condensr.com
Open Text Summarizer	Multi document – Multi lingual	Downloadable from http://libots.sourceforge.net/
Copernic's Summarizer	Multi document – Multi lingual – Commercial	Trial version available at http://www.copernic.com/en/products/summarizer/index.html
Intellexor's Summarizer	Multi document – Multi lingual – Commercial	Trial version available at http://summarizer.intellexer.com/
Essential Summarizer	Multi document – Multi lingual – Commercial	Trial version available at https://essential-mining.com/es/produits.jsp?ui.lang=en
SSSearch	Multi document – Multi lingual – Commercial	Trial version available at http://www.kryltech.com/summarizer.htm
Microsoft Word Auto Summarize	Multi lingual – single document summarizer	Available as a function in the Word application
Centrifuser	Domain and genre-specific multi document summarizer	Available from http://www1.cs.columbia.edu/nlp/tools.cgi
QCS	Query, Cluster and Summarize Multi-document	Online demo with limited access http://stiefel.cs.umd.edu:8080/qcs/index.html

Table 2.1: Some of the public summarization systems available on the web and in common applications

2.4.1 MEAD

MEAD [102] is a multi document extractive summarizer that scores sentences according to a linear combination of features including centroid, position and first sentence overlap. These scores are then refined to consider cross-sentence dependencies, chronological order and user supplied parameters. Initially, documents are segmented into clusters with a distinctive theme covering each cluster. Then, all input documents are represented with TFIDF vectors. Other features are also factored in at subsequent stages to help assign a score to each sentence. The overall score S_i of a sentence i is computed as the weighted sum of the considered features as follows:

$$S_i = w_1 * C_i + w_2 * F_i + w_3 * L_i \qquad \textbf{2.1}$$

Where C_i is the similarity scores between sentence i and the cluster theme it belongs to, F_i is similarity score between i and the first sentence in the document it belongs to and L_i is the position score for sentence i. w_1, w_2 and w_3 are the weights assigned to each feature. After initially computing S_i, the sentence i is further re scored to take into account the redundancy in the summary. Because all documents are modelled as BOW, the summarizer is multi lingual and domain independent. Further details about the summarizer and its implementation can be found in [102], [103] and [104]. It should be noted here though at least two of its features give higher weight to sentences at the beginning of document. This makes the summarizer most suitable for news articles where authors tend to include most important sentences at the beginning.

2.4.2 Newsblaster

Newsblaster is a multi-source multi lingual summarization system [105]. It has an online demo which helps users finds the news they are most interested in. It crawls the web to read news articles from different sources, clusters and categorizes these articles and provides an update for each cluster. It uses different types of summarizers on the collected articles clusters depending on the detected articles types. For example, single-event articles are summarized by integrating machine learning and statistical techniques to identify similar sentences across the processed articles [106]. It also uses a cut and paste method for extracting important phrases from sentences and adding them to the summary [105].

2.4.3 QCS

Given a query, the Query, Cluster and Summarize (QCS) system [107] separates the retrieved documents into topic clusters and creates a summary for each cluster. LSA is used for documents retrieval, spherical k-means for clustering and a HMM-based module for extractive summarization. The system has an online demo with limited access to only the DUC collection dataset and MEDLINE documents.

2.4.4 MASC

MASC [108] is a multi-document summarizer that generates Multiple Alternative Sentences Compressions (MASC), instead of unaltered source sentences, as candidate summary components. It uses weighted features of the candidates to select candidates and construct summaries. MASC differs from MEAD and many other summarizers in that multiple variants of a single source sentence are available to the sentence selector to choose for inclusion in the summary. The system for this summarizer was built on top of two other variants which used different techniques for compressing sentences. One is called HMM Hedge [32] which uses a noisy channel model with language models of newspaper stories and headlines to generate the most probable compression for a source sentence. The second is called Trimmer [24] and uses syntactic rules to compress sentences. The summarizer is mono lingual and can only be applied to one language, English, due to the syntactic rules it applies and the language-dependent models built for compressing sentences.

2.4.5 Condensr

This system provides extractive multi-document summaries and sentiment analysis [109]. It leverages the documents structure along with cue words and phrases and contextual information to build an HMM-based model that also aids with summarization. It is designed to primarily handle reviews and has an online demo for summarizing restaurants reviews and viewing the reviewers' sentiments.

2.4.6 Open Text Summarizer

The Open Text Summarizer is an open-source tool that analyzes texts in various languages and tries to present the most important parts of the text and present them in a summary. It works by first removing stop words from the text and stemming all terms. Then, a weight is assigned to each word based on its frequency and sentences with highest weighted terms are chosen for the summary. It has a downloadable version in addition to an online one. In addition, it ships with several Linux distributions such as Ubuntu and Fedora.

2.4.7 Commercial Summarizers

Copernic's summarization software is multi lingual and available commercially on the company's website. It claims to use statistical and linguistic algorithms to provide extracts-based summaries. Intellexor's Summarizer is another commercial application. It claims to create theme-oriented, structure-oriented and concept-oriented summaries. Microsoft's Auto Summarize feature in its common Word application provides a multi-lingual single document summarization. The Essential Summarizer provided by Mining Essential is a cross-lingual multi-document summarizer. It covers 20 languages and is able to provide translated summaries in a language different from the input documents language. The summaries provided by the system have sentences with varying degrees of font size to illustrate the importance of its sentences. Sentences with bigger font size are more essential and important than others with smaller font sizes. Subject Search Summarizer (SSSearch) is yet another multi-lingual and multi-document commercial summarizer.

2.5 Limitations of Current Approaches

Many of the summarization systems developed in previous work and mentioned above primarily rely on the words present in the documents text. Unless a background repository is being used, the system is always limited to the words explicitly mentioned within the provided text. In BOW-based systems where training takes place, other limitations appear. One limitation is ignoring the words that appear in the testing documents but *not* in the training documents. Due to the design of such systems, they simply lack the ability to analyze such words. In some cases, these words appear in abundance in the training set, but very infrequently in the testing documents. This also gives a similar effect and the system treats these words as unimportant.

Another limitation in the BOW approaches is the lack of detection for the implicit relationships between words in a document, or the training set. Without the ability to find the similarity and relatedness between terms such as "Petroleum" and "Oil", the systems would treat these terms as two separate unrelated entities and this may affect the judgment of the their importance in a document. The ability to detect such implicit relationships between terms within a document requires an external knowledge and an analysis module to learn the relationships between the different terms. BOW systems also are affected by a similar limitation in the detection of *concepts*. For example, with the appearance of phrases such as “oil extraction”, “oil production”, “petroleum production” and “oil exploitation” within a document, the system should be able to tell that these phrases refer to one single concept. The relatedness between detected concepts is not explored in BOW systems. Entities such as “Hiroshima”, “Nagasaki” and “Atomic Bombing” should be found to be related if all appeared within a document or a set of documents.

In Bayesian-based summarizers such as the one developed by [22], probabilistic measures are applied to link words within a document with each other. This however, still does not make use of new information outside what’s already available in the training and testing documents. Trained and semi-supervised systems such as [110] and [111] employ the co-occurrence and correlation of information within the training documents and those in the test data. Again, no new information outside what is already available in the test documents is utilized.

Different Latent Semantic Analysis (LSA) [112] models were employed in previous systems. With LSA, large corpora were analyzed to extract the main concepts using Singular Value Decomposition (SVD). The extracted concepts were used to help embed new features in the summarization system. The new features were then used in addition to those extracted from the test documents as in the work of [113]. It was found in earlier work that LSA helped in various applications such as Text Classification [114] and Summarization [88] especially when the training data is scarce. However, it was noted in subsequent work [115] that the use of the virtual concepts introduced by LSA can degrade the performance of the system it is employed in. The repositories I employ in the work of this thesis contain real concepts which have been chosen and defined by humans.

As described in the previous sections, when human beings summarize a document they usually read the text first, understand it and then attempt to write a summary. In essence, these processes require at least some basic level of background knowledge by the reader. At the very least, the human would have to understand the Natural Language that the text is written in. This should allow the reader to understand the words of the documents according to the text they are placed in. In this thesis, an attempt is made to bridge the gap of machines understanding by providing a system with knowledge repositories constructed by humans and containing real human concepts. The usage of knowledge repositories which have reasonably wide coverage should allow the system to be applied to documents from a wide range of areas that other summarizers can not be effectively applied to. The framework implemented and described in this thesis is generally unsupervised and does not require training once the required features from the knowledge repositories are constructed and built, giving it an edge over other supervised systems that require training.

Most of the summarization systems described in the previous sections are extractive. They select sentences for the summaries from only what exists in the input documents or possibly a single compressed version of them. In essence, this leads to inefficiencies when dealing with summary sentences redundancy, diversity and coverage. This is especially evident when noting that no compression tool that produces only a single compressed version of sentences can provide the best sentence for every context. When adding an extracted sentence to a summary, the redundant information in the sentence that already exists in the summary should aid with the sentence compression decision. In this thesis, I adopt a Sentence Simplification Module (SSM) that produces multiple compressed versions of each candidate sentence. Before adding a sentence to the summary, only the best and least redundant version of a relevant and important sentence is added.

2.6 Text Summaries Evaluation

When designing and updating a summarization system, it is necessary to have a tool or method to track the performance of the system and the changes being made. For summarization, there are two main types of evaluations: Intrinsic and Extrinsic. Intrinsic evaluations are used for evaluating the quality of summaries. They may help in answering questions about a summary such as its coherence, grammar and whether it suggests incorrect information deductions from the original text, or redundant information within the summary itself. Extrinsic evaluation on the other hand helps in answering whether a summary meets the purpose it is generated for. For instance, it can help determine whether a summary is a good replacement of the original documents and conveys the most important information within it.

2.6.1 Intrinsic Evaluations

Intrinsic evaluations base their judgments on the output of the summaries. Human intrinsic evaluations measure the clarity, cohesion and informativeness of a summary [116]. Automatic intrinsic evaluations compare systems summaries with other reference summaries generated by humans. Among the main intrinsic tools used are Precision/Recall, BLEU, ROUGE and Pyramid. The Pyramid requires manual annotation of systems summaries and reference summaries before they can be automatically compared. Precision/Recall, BLEU and ROUGE on the other hand are fully automated and only require reference summaries.

Precision, Recall and F-Measure are among the simplest evaluation approaches available that measure the relevance of a summary by the relevance of the sentences it contains. Precision (P) is the number of sentences appearing in both the system summary and the reference summary divided by the number of sentences in system summary. Recall (R) is the number of sentences occurring in both system and reference summaries divided by the number of sentences in the reference summary. F-Score is a composite combining both P and R [12]. The F-Score can be computed with the following formula:

$$F = \frac{(1+\beta^2)PR}{\beta^2 P + R} \qquad \textbf{2.2}$$

Where β is a weighting variable that is adjustable to affect precision and recall.

The Precision/Recall measure is not without its limitations. Suppose that two persons were asked to choose the top two most important sentences from a document to represent as a summary. It is possible that the human judges could choose two different sentences that carry the same meanings. I can name these summaries A and B. If one of the two human summaries, say A, was used as a reference summary, a system summary producing sentences that exist in summary A and not B would rank highest. If another system produced a summary containing sentences from summary B, the first system would still rank higher.

Bilingual Language Evaluation Understudy (BLEU) [117] is an n-gram precision based system originally developed for machine translation. It works by finding the number of n-grams in the system summary that matches those in a reference summary. BLEU is a precision-based evaluation giving higher scores to system summaries with content that appear the most in reference summaries. Because of this, it rewards systems that rarely include incorrect sentences in their summaries. On the other hand, it does not penalize systems that do not include the right sentences in their summaries.

As a response to address the limitations of the BLEU system mentioned earlier, the Recall Oriented Understudy for Gisting Evaluation (ROUGE) was proposed in 2003 [118] at the Information Science Institute. It is roughly based on BLEU but focuses on recall instead. Also, it measures words overlaps in sequences and was found to correlate better with human evaluations than many other systems.

Several variants of ROUGE have been proposed [119]:

- ROUGE-N: counts contiguos n-gram. N ranges from 1 to 4.
- ROUGE-L: Longest Common Subsequence (LCS) based metric
- ROUGE-W: Weighted LCS favouring sequential LCS
- ROUGE-S: uses skip-bigram: words pairs in sentence order, ignoring gaps
- ROUGE-SU: uses skip-bigram and unigram.

In the Documents Understanding Conference (DUC) and Text Analysis Conference (TAC), ROUGE-SU and ROUGE-2 have been the standard metrics for summaries evaluation. With ROUGE-SU, sentences such as “*The truck was bought by the company*” and “*The company bought the truck*” are found to be a match since the order of the terms is not relevant as long as they are within the same sentence. With ROUGE-S, they would not be a match.

In contrast to ROUGE and BLEU, the Pyramid method [120] requires humans marking and grouping items within summaries. It attempts to address the issue that summaries generated by summarization systems may include information which are related to the topic but not necessarily included in the reference summary. Every reference summary is first annotated with Semantic Content Units (SCU) which are facts or important events at the clause level. The SCUs are then given weight based on their occurrence frequencies within all the reference summaries and are organized in a Pyramid based on their weights. SCUs appearing near the top of the pyramid are of higher weights and appear more frequently in the reference summaries than SCUs near the bottom. The SCUs that appear in all the reference summaries will be in the top of the pyramid while those appearing in single reference summaries are in the bottom of the pyramid.

The advantage of using the Pyramid method is that, unlike n-gram based methods, it is based on the semantic content of the summaries. However, due to it being labour and time intensive, it is not being used frequently with many systems especially when the summarization system is in a continuous update stage incurring many changes at different stages in time.

For evaluating multi-lingual summaries, a framework that relies on similarity measures was suggested in [121] to evaluate English and Chinese summaries. The framework is capable of evaluating single, multi-document, abstractive and extractive summaries. As for speech summarization, custom measures such as the Summary Accuracy in [122] were proposed. Other measures such as precision, recall and ROUGE were also used in other systems for speech summarization.

2.6.2 Extrinsic Evaluations

Extrinsic evaluations [123] help show how well a summarization system performed in a specific task. They can help measure the relevancy of a summary to a topic or indicate a category for a document. When choosing and implementing a specific task, it is important that the task is clear enough for any person to attempt and answer with high level of confidence. If a task is not clear enough and persons can not agree on the answer, it is not possible to use the task and its implementations to evaluate systems summaries. Question answering, Information Retrieval and relevance judgment are examples of tasks that can be used to test systems and evaluate their performance. An example for this is what was performed in the TIPSTER SUMMAC Text Summarization Evaluation [4] in which 16 systems participated. One of the main goals of the evaluation was to judge the usefulness and relevancy of the information presented in the participants summaries.

2.7 Conclusion

The purpose of this chapter is to give an overview on Text Summarization and a review of previous work in the field and how they relate to what is provided in this thesis. Sections 2.1 and 2.2 provided some basic notions on Text Summarization and Automatic Text Summarization. Related work and State of the Art summarizers were given in section 2.3. Section 2.4 illustrated examples of online and public commercial and academic summarizers. Then, I presented the limitations of current approaches in section 2.5 . In section 2.6, I described how evaluation is performed in the literature for different summarization tasks.

Due to the interdisciplinary nature of the task of summarization, several disciplines are involved including Philosophy, Psychology and Logic. Special emphasis was given in this chapter on Artificial Intelligence and NLP-related methods and summarizers. While the methods and techniques described in the related work section are not exhaustive, they still provide the main themes of the NLP-based summarizers used in the literature and the main limitations of the current approaches. Among the main limitations mentioned is the dependency on methods that solely rely on the explicitly mentioned terms within the training and test documents. In my work, I use external repositories containing human knowledge to aid with the summarization task and help infer the hidden relationships between the different concepts in a document. Before using the repositories, it is necessary to define what features will be used from the repository, how to extract and build these features, and then how they can be used in the summarization task at hand. In the next chapter I give an overview on these processes and how they are applied with the two repositories I employ, namely WordNet and Wikipedia.

Chapter 3

Features Generation and Selection

In the previous chapter, different methodologies have been discussed and their limitations have been highlighted. In this chapter, I describe how the developed methods in this thesis select and generate features from external knowledge repositories which address the mentioned limitations.

3.1 Overview

Feature Generation is the process of building and constructing new features from those available in a given text [124]. Generating features can be especially useful when the supplied information with text documents is limited or reasoning is required. The methods used in this thesis utilize external repositories to aid in generating and selecting features from text. The generated and extracted features can be in different forms such as Named Entities (NE), Time Expressions (TE), Part-of-Speech (POS) tags and Concepts. In a preprocessing step, these features are specified, prepared and built in what I call the Features Generator (FG). During the analysis of text documents, the FG is used to augment or replace the extracted bags of words. After assigning a weight to each feature, filtering takes place to determine the most influential parts to be applied to the application at hand, whether it is Text Summarization, Word Sense Disambiguation or Text Classification.

To illustrate the need for features generation and construction, take the example of video recording where a frame is captured by a camera typically every few milliseconds. Each frame is represented by a features vector for a single view and can not alone create a video. Collectively at the macro-level, all frames can form a higher and richer view in the form of a video when combined. The case with features extraction and construction for text processing is very similar. I use external knowledge repositories to construct unique and rich text representative features vectors that when combined in a specific task at the macro level can lead to better representation and performance of the system.

An overview of the FG and its role in Automatic Summarization is shown in Figure 3.1. It can be noted that the figure is split into two segments, namely A and B. Segment A shows the processes being performed and applied on the information repository at hand. Depending on the repository type and the algorithms applied to extract features, the processes are usually performed once offline and the results are stored for subsequent use, and are called Generated Features. Processes in segment A do not require an interaction with testing documents or training data. In contrast, segment B modules are fed with the Test documents to be summarized and produce the output summary. If the system relies on training documents, they are also fed to the parser. The Generated Features from segment A may be used to aid in parsing the documents in some systems depending on the methods employed and the algorithms applied. This optional data flow is represented with the dotted arrow line in Figure 3.1. The generated features are, however, part of the inputs required for the analyzer and are given weights and filtered within that module as will be described in the next two chapters.

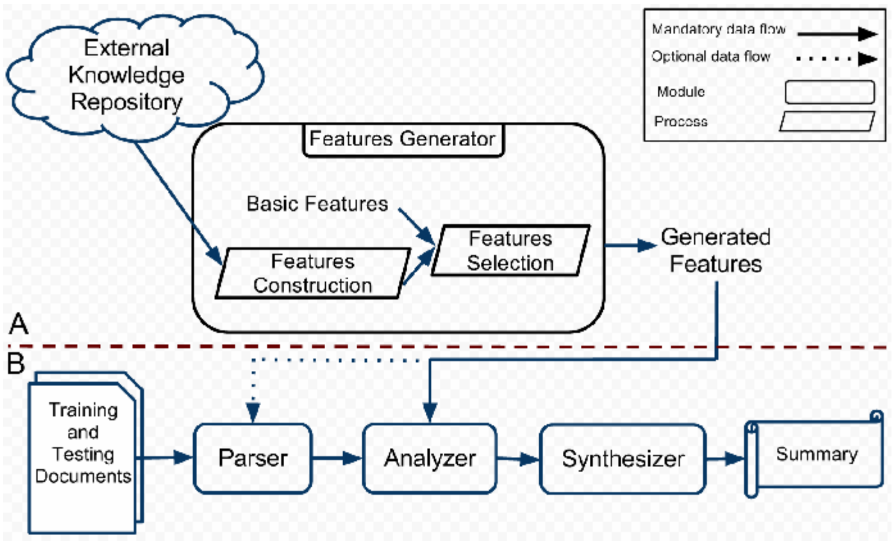


Figure 3.1: Overview of the Features Generator and its role in Summarization

3.2 The Need for a Suitable Repository

One of the main goals for using external repositories and generating new features is to be able to apply some reasoning on a text document. Reasoning, after all, about the different concepts in a natural language is a common human task. As described in the previous chapter, researchers have been working for decades to supply machines with similar capability. In my thesis, an attempt is made in a similar direction. I use external repositories to generate features that enable machines to apply reasoning by measuring semantic distance between different human concepts.

Semantic Distance is a generic measure used to define how close or distant two units of text are in terms of their meanings [125]. The units of text can be words, groups of words, sentences or paragraphs. The text units may sometimes be referred to as concepts. As an example for two units which are close in their meanings are *Apple* and *Watermelon*. They are both fruits containing seeds and edible. However, the word *Apple* in a sentence such as "*Apple released the latest Mac last June*" carries a different meaning since it refers to the company *Apple* and the not the fruit. Thus, it can be concluded that a term meaning varies based on the context it is placed in. The same can be said about semantic distance between different units of text.

It is therefore possible to outline a set of rules that should be met in the repositories utilized for FG. First of all, the repository should contain *Text Units (TU)* or *Concepts* defined by humans. Each TU or concept should have a corresponding meaning understood by humans when placed in a specific context. Secondly, each TU should have its different meanings clearly defined and distinguished from the rest based on the context it appears in. As shown in the previous example, the word *Apple* may refer to the company or the fruit. Third, it should be possible to induce the semantic distance between the different TU and their different meanings. For example, it should be possible to quantify the relation between the fruits *Apple* and *Watermelon* and say they are more related than *Apple* and the animal *Lion*.

The above-described rules do not have to be directly mentioned or explicitly existent within the repository. As long as a method exists to adapt or induce the missing information, the repository may still be suitable for the tasks at hand. After choosing the right repository, a mapping algorithm is applied within the parser and/or analyzer to match the extracted TUs and Concepts to the test documents at hand.

Several repositories exist which meet the mentioned requirements. Some of them are domain-specific while others are more generic. Some were prepared by human experts in different fields (closed repositories) while others were the result of a joint effort by the web community (open repositories). Among the common closed repositories is WordNet which has been used to enrich text documents for different types of applications including classification, summarization and categorization. Another example for one of these repositories is the general-purpose Ontological Semantic (OntoSem) [126] repository. It was built mainly by human acquirers using interactive tools. The ACM Computing

Classification System (CCS) [127] ontology is another example of a closed ontology built and maintained by experts. It undergoes periodic updates and redesigns but always seems to be out of date as a classification of computer science concepts. It has gone through six revisions with the first version being published in 1964, and then it was revised in subsequent versions in 1982, 1983, 1987, 1991 and the latest in 1998. The Medical Subject Headings (MeSH) is yet another domain-specific ontology defining over 18,000 categories and is linked to scientific articles. A shortcoming shared by all closed ontologies is the costly task, in time, labour and other resources, of designing and building it at first. After being built, the ontology would need to be continuously maintained and updated to reflect the addition and evolution of concepts. CYC [128] for example, which has been under continuous development and maintenance by experts for almost two decades, still suffers from incompleteness and incomprehensiveness. Its aim is to contain all common sense knowledge which an average adult person should already know, but it is not intended to cover people's general information needs. It has been estimated by its author that it would take 350 man-years of effort to complete the CYC project [129]. It has a smaller open-source version called OpenCYC[4].

The open ontologies on the other hand which are usually created and maintained by the web community, have the advantage of being more up to date. Wikipedia is an example of such large-scale knowledge repositories. It has been developed and maintained by the web community and has considerably good accuracy surpassing some of the experts-made ontologies [130]. The Open Directory Project (ODP) and Wiktionary are other open ontologies. While the breadth of the open ontologies exceeds that of the closed ones in most cases, their format and structure are not as easy to handle. This is due to them being

[4] http://opencyc.org/

developed from the start to be browsed by web surfers and without the intent of being used in NLP applications.

With the different types of available ontologies, the NLP community always had to deal with several issues. First, it had to choose the most suitable ontology for the task at hand depending on the scope and the domain of the ontology and the targeted application. Second, an understanding of how the concepts within the ontologies are defined, what they mean or represent and how they are used by humans is important. For those concepts which overlap in their meanings with others or share the lexicons, it may be important to differentiate between them or define a degree of relevancy among the concepts. Third, it is necessary to decide how to map the concepts or entries existing within the chosen ontologies to text documents. Optimally, the degree of relevancy would be declared during the mapping process.

In my work, the methodologies that have been developed rely on two repositories: the first is the hierarchically-structured repository that was created by linguistic experts and is rich in its explicitly defined lexical relations: WordNet. The second is the open-World knowledge ontology Wikipedia. In the next section, I give an overview on each repository and highlight how the semantic distance is being computed with the features extracted from each. Also, I describe how the mentioned issues are being addressed by my use of the chosen repositories.

It should be noted that with WordNet, I use the terminology *Semantic Similarity* to describe how close in meaning two TUs or concepts are while *Semantic Relatedness* is being used with Wikipedia. Both are two types of Semantic Distance and may have been used in the literature interchangeably in certain contexts [125]. However, the former is in fact a subset of the later. To illustrate this, I give the example shown in Figure 3.2. *Apples* and *Oranges* are similar while *Apples* and *Seeds* are related. On the other hand, *Apples* and *Lions* are unrelated and not similar.

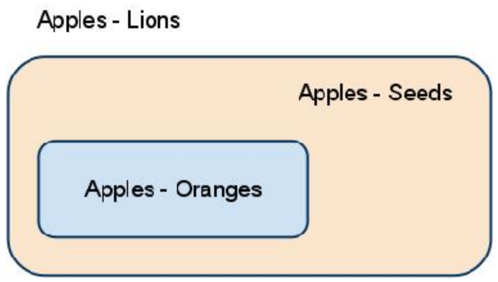


Figure 3.2: An Example showing that Relatedness is a subset of Similarity

3.3 Using a Hierarchically Structured Repository

WordNet is the product of the research project that was conducted at Princeton University to create a model of a native speaker lexical knowledge and store it in a machine-readable dictionary [131]. It contains open-class words of type nouns, verbs, adjectives and adverbs. Other closed-class words of type pronouns, prepositions and conjunctions are excluded. In the next subsections, I give an overview on WordNet first describing its main aspects and structure. I follow by outlining how semantic similarity has been computed in the literature using WordNet.

3.3.1 WordNet

Words in WordNet are organized in synonymous group sets called *synsets* and they form the basic structure of WordNet. A single word may carry more than one meaning and each meaning is given an entry in WordNet called a *word sense*. For example, the word *plane* may refer to *aircrafts with wings* or *the unbounded two dimensional shape in Mathematics*. A synset contains synonymous word senses to help describe a word. For example, the former meaning of the word plane is represented with the synset *{airplane, aeroplane, plane}*. Each synset may have a gloss describing the meaning of the word. For example, the gloss for the synset *{airplane, aeroplane, plane}* is “an aircraft that has a fixed wing and is powered by propellers or jets”. Many synsets come with associated usage examples such as “*the flight was delayed due to trouble with the airplane*” for the mentioned synset.

Senses in WordNet are ordered according to their usage frequencies with the most common sense being in the top of a senses list. The usage frequency is determined based on the number of times a sense is tagged as used in various semantic concordance texts[5] [132]. If some senses are not tagged, they follow the tagged senses in their order. If all senses of a word are not tagged, the appearance order is random. WordNet words are usually represented in a specific format with each word tagged with its Part-of-Speech (POS) and sense number. Four letters are used to represent the four available POS types in WordNet: *n* for nouns, *v* for verbs, *a* for adjectives and *r* for adverbs. For example, the first sense of the word *plane* is represented as *plane#n#1*.

Version 3.0 of WordNet contains 155,287 words and 117,659 synsets. The number of Word-Sense pairs is 206,941. The majority of the words are nouns with a count of 117,798. The number of verbs is 11529, while adjectives and adverbs are 21,479 and 4,481 respectively.

Words senses and synsets are connected via a variety of relations. The relations connecting words senses are called *Semantic Relations* while those connecting synsets are *Lexical Relations*. For example, nouns have the following semantic relations:

- Hyponym/Hypernym (IS-A , HAS A)
- Meronym/Holonym (Member-of, Has-member)
- Meronym/Holonym (Part-of, Has-Part)

[5] A semantic concordance corpus is a textual corpus and a lexicon so combined that every word in the text is linked to its appropriate sense in the lexicon.

- Meronym/Holonym (Substance-of, Has-Substance)

In Figure 3.3, a fragment of WordNet hypernyms/hyponyms is shown. It can be noted that {land, dry land} is the hypernym of {island}. In the same time, {island} is a hyponym of {land, dry land}. For verbs, other relation types exist such as troponyms and its inverse hypernyms. {travel, go} is a troponym of {fly} and {fly} is a hypernym of {travel, go}. These relations form taxonomies with tree-like structure. All nouns and verbs synsets belong to taxonomies. Some synsets belong to a single taxonomy while others belong to more.

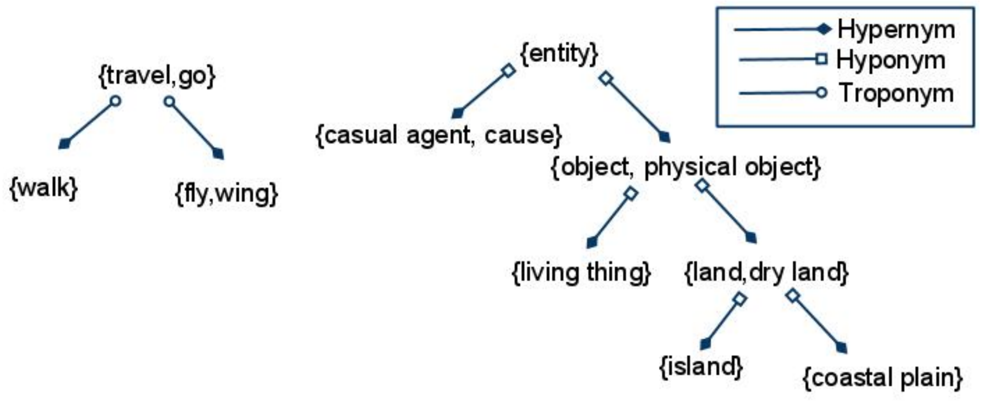


Figure 3.3: WordNet Hypernyms, Hyponyms and Troponyms

There also exists other lexical relations such as antonyms and derived-from. Lexical relations are between word senses and not necessarily synsets. For example, the sense dark#n#4 belongs to the synset {night#n#1 , nighttime#n#1 , dark#n#4}. The sense night#n#1 has an antonym relation with day#n#4 while the sense dark#n#4 does not even though night#n#1 and dark#n#4 both belong to the same synset.

3.3.2 Semantic Similarity

Several measures have been defined to quantify the Semantic Similarity between any two senses. Some utilize only the taxonomies within WordNet and the relations defined between its units while others are driven by data derived from different types of analysis on external text corpora. Some others try to merge the two in their methods. In [133], the authors define the distance between two concepts in an IS-A semantic network as the length of the shortest *Path* connecting two nodes. The distance can then be used to find the semantic similarity between any two synsets s_1 and s_2 by applying the formula:

$$Sim_{Path}(s_1, s_2) = \frac{1}{dist(s_1, s_2)} \qquad \textbf{3.1}$$

For domain specific applications with highly constrained taxonomies, this *Path*-based method produced acceptable results [134]. It is, however, not suitable when the links density in a taxonomy is not uniform. In WordNet, different parts of the network hierarchy have higher densities than others. For example, some nodes in the *plant/flora* segment of WordNet may have several hundred child nodes. This great density of links within that segment may suggest closer distances between its nodes [135]. Also, the *Path* method does not utilize other relations that exist within WordNet such as antonyms and holonyms. In [136], *Resnik* proposed a measure in an IS-A taxonomy utilizing Information Content (IC) to compute the semantic similarity. *IC* is the information content of a synset *S* calculated from some corpus and is computed as:

$$IC(S) = -\log(P(S)) \qquad \textbf{3.2}$$

where *P(S)* is the probability of *S* in the used corpus. In *Resnik*, the semantic similarity between two concepts depends on the amount of information shared between the two. Formally, this is computed with:

$$Sim_{res}(s_1, s_2) = \max_{C \in S(s_1, s_2)} IC(C) \qquad \textbf{3.3}$$

Where *S(s₁,s₂)* is the set of concepts that subsume both s_1 and s_2, and *C* is a concept subsuming both s_1 and s_2. Jiang and Conrath in [47] pointed out that using both *IC* and the taxonomy structure is superior to using either. Thus, they proposed the following formula:

$$Sim_{jcn}(s_1,s_2) = (IC(s_1)+IC(s_2)) - 2\times Sim_{res}(s_1,s_2) \qquad \textbf{3.4}$$

Lin in [48] extended Resnik's measure by also utilizing the information theory and produced a measure expressed by:

$$Sim_{Lin}(s_1,s_2) = \frac{2\times Sim_{res}(s_1,s_2)}{(IC(s_1)+IC(s_2))} \qquad \textbf{3.5}$$

All of the methods mentioned above have one thing in common: they all utilize the IS-A hierarchy within WordNet. Some use other external data such as IC, too.

In my work, the focus was on using Jiang and Conrath (JCn) when computing the semantic similarity between any two concepts. This measure was found to correlate best with human judgements when compared against the rest of the common semantic similarity measures [137]. The feature generator described in the previous section represents concepts or Text Units as vectors and each is labelled by its most distinguishing words, or attributes. Mapping pieces of text to their corresponding concepts is the main task that would result in the generated features. When using WordNet as the backend repository, the concepts are viewed as the different words senses in WordNet. Determining the right sense will depend on the context of the text and is described in more details the next chapter.

The methodologies described in the previous section that rely on WordNet are not without their limitations. This is in part due to the taxonomy structure of WordNet being limited. For example, coverage for the relations between physical entities and abstract concepts is not at the same depth level as desired. The semantic similarity between *Nose* and *Smell* for instance is much less than that of *Nose* and *Foot* because *Nose* and *Foot* are both hyponyms of Entity while *Nose* and *Smell* are hyponyms of Abstraction. In the next section, I describe Wikipedia, the second repository I used, and how the Features Generator interacts with it.

3.4 Using Open-World Knowledge

When humans summarize a document, they usually attempt to understand it first. This requires an understanding of the language the document is written in. Also, it may require that the summarizer has background knowledge about the concepts mentioned within the document. When machines face a similar task, it is necessary to take the mentioned human factors into account. Machines need to be supplied with background knowledge, and the best suitable source for this is an encyclopaedia. This is supported by the breadth hypothesis proposed by Lenat in [138] in which he says "*to behave intelligently in unexpected situations, an agent must be capable of falling back on increasingly general knowledge*". However, the use of encyclopaedia presents yet another set of challenges. First, using the textual data available in encyclopaedia requires natural language understanding. In addition, common sense may also be required for understanding text documents, especially for humans [3]. In an attempt to address part of the problem, Lenat started the CYC project to create a repository of common sense knowledge of human beings. The aim of the CYC project is to create a repository containing all common sense knowledge an adult person would have. It is not its purpose to resolve people's information needs. As mentioned earlier, the author estimates that 350 man-years are required to complete building the repository. A smaller version of the repository exists in an open-source form and is called OpenCYC but it still suffers from the same limitations.

In this thesis, I attempt to use the largest encyclopaedia known to date [139], Wikipedia, in the task of Automatic Documents Summarization.

3.4.1 Wikipedia

Wikipedia is known to be the largest available, fastest growing, and most recent encyclopaedia. It is hosted and funded by the Wikimedia Foundation[6], a non-profit organization which hosts some other related projects such as Wikibooks and Wikinews. Its articles, over 15 million, are written, revised, updated and maintained by over 153,000 volunteer editors and it spans over 240 languages. Its nearest competitor, the Britannica Encyclopaedia, has been in development since the 1700s and has approximately 120 thousand articles[7], which is orders of magnitude less than that of Wikipedia.

The articles in each language vary in quantity ranging from few pages to 3,289,927 pages for the English version[8]. An *article* can be seen as the basic unit in Wikipedia describing a single topic thoroughly while being constantly revised and updated causing its depth and breadth to increase with time. The continuous updates and revisions to articles give Wikipedia a unique adaptability feature allowing it to reflect the most recent major events or concepts.

The issue of Wikipedia’s accuracy has captured the interest of Media and many researchers. In a study [140] that conducted an experiment to compare some Wikipedia articles against their Britannica counterparts by academics, it was found that subtle errors exist in both such as omissions and misleading statements. However, the study concluded that Wikipedia approaches the accuracy of Britannica. In [141], some chosen Wikipedia articles were compared against their equivalents in the Medscape Drug Reference and

[6] http://en.wikipedia.org/wiki/Wikimedia
[7] http://en.wikipedia.org/wiki/Wikipedia:Size_comparisons
[8] As of writing this on January 2011

found no factual errors. However, it was noted that Wikipedia found only 40% of the addressed questions while their experts-made counterpart found 83% which hints that Wikipedia has an omission problem. However, as mentioned in the more recent study in [142], Wikipedia was still found to compare favourably against all other sources which people would turn to if Wikipedia did not exist and the strengths it has outweigh its weaknesses.

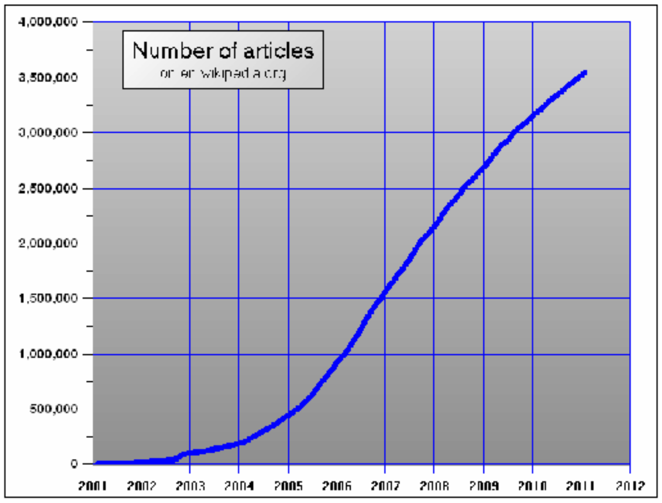


Figure 3.4: the growth rate over time of the English Wikipedia[9]

Each article in Wikipedia contains *text* describing the topic of the article along with links (internal and external) to other relevant topics. The aim of the links is to provide the readers with insight and additional information about other relevant topics. In addition to the text describing the article, each article is uniquely labelled with a set of terms forming a *title.* When two or more articles discuss a topic with a word carrying more than one meaning, the title is usually augmented with a descriptive term to differentiate between the two articles. For example, the term *bar* carries more than one meaning and each meaning has an associated article with a unique title describing that meaning as in *Bar (establishment)*, *Bar (unit)*, and *Bar (music)*. Augmenting a title with keywords to

[9] From http://en.wikipedia.org/wiki/History_of_Wikipedia

differentiate the different senses of a term is not always the case as evident for the topics titled *Tree*, and *Tree (data structure)* where the former refers to the tree plant while the later refers to the computer data structure. This can be viewed as an inconsistency within Wikipedia caused by the participation of large number of editors carrying different opinions.

Another aspect worth mentioning in Wikipedia is the disambiguation pages which have been created for ambiguous terms carrying more than one meaning. The disambiguation pages provide links to different articles each describing one meaning of the term. The title *Rice (disambiguation)* for instance is the title of the disambiguation page for the term *Rice* listing links to different articles providing different meanings for the word. There also exist redirect links which simply provide alternative terms describing the same topics as the one existing within Wikipedia. The purpose of forming the alternative names in the redirect links is to highlight alternative names, abbreviations, shortcuts, alternative spellings, or likely misspellings. For example, the article titled *United Kingdom* has the redirect link *UK* pointing to it.

One more aspect in Wikipedia is the categories and their overlapping trees structure. Every article is assigned to one or more categories which it belongs to. Every category belongs to one or more parent categories and can contain subcategories. There is one top-level category named Contents which only have subcategories but no parents. All other categories are below this parent category. The whole structure of the articles and their categories in Wikipedia can be viewed as a directed acyclic graph.

3.4.2 Semantic Relatedness

In Wikipedia, just like other encyclopaedias, articles exist describing a variety of topics. Each article can be viewed as a *concept* and is attached to a body of text (the article content) describing the article’s main topic. The articles and their content are in the same form as the documents to be summarized since they are all text. The use of a devised semantic similarity measure allows for augmenting text documents with the features extracted from Wikipedia and its large amount of world knowledge. In effect, this replaces the need for understanding the actual content of text documents and allows bypassing the difficulties highlighted above. Take the concept “Lion” as an example. One way to describe it is by the definition “*large gregarious predatory feline of Africa and India having a tawny coat with a shaggy mane in the male*” as given in WordNet. Another way is to say that it is strongly related to “*Big Cat*”, “*Scavenger*”, “*Felidae*” and “*Mammal*” and is less strongly related to “*Tiger*” and “*Leopard*”.

The goal of the Wikipedia Feature Generator is to enrich the representation of a text document by augmenting it with features extracted from Wikipedia. The features include the detected concepts and the relevancy between them within a document and others within the same document set (in multi-document summarization tasks for example). Each detected concept is represented with an attribute vector whose elements are all the other concepts and the degree of relatedness between each and the main concept. In the vector list, concepts with weak association or rather small relevancy degree are removed from the list. To compute the relevancy degree between all concepts, I use the features extracted from Wikipedia for the task including the articles titles, redirect links, articles content, articles categories, and articles links. Figure 3.5 gives an overview of Features Generation for a Wikipedia article titled “Mouse”. More details about the methods used and the implementation details are presented in chapter 5.

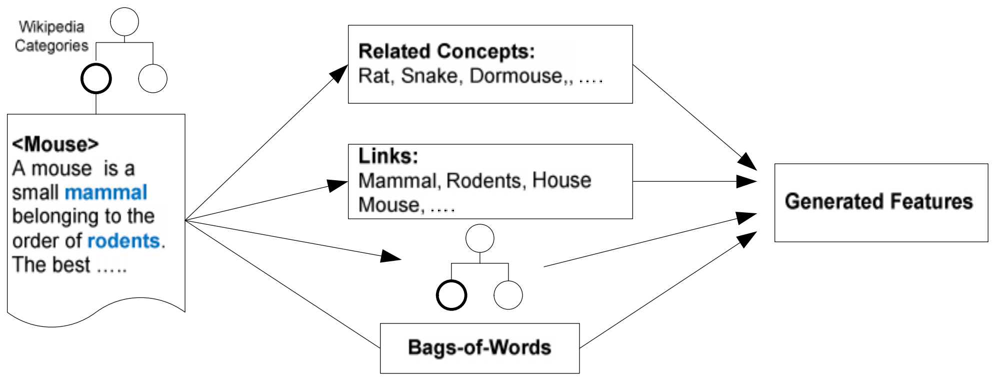

Wikipedia Categories
<Mouse>
A mouse is a small mammal belonging to the order of rodents. The best …..
Related Concepts:
Rat, Snake, Dormouse,, ….
Links:
Mammal, Rodents, House Mouse, ….
Bags-of-Words
Generated Features

Figure 3.5: an Example for Wikipedia Features Generation

It should be noted that filtering and preprocessing is first applied to the used Wikipedia dump. Although there are 3.5 million content pages in the English version of Wikipedia, they are not all with the same importance. Some of the articles are too short, while others contain only statistical data and tables or dates. The developed filtering module applies a set of rules to ensure that all concepts used in any task are attached with rich text contents.

3.5 Summary

Different methodologies and techniques have been mentioned in the previous chapter. The methods proposed in this thesis rely on the use of external repositories containing human knowledge. These repositories can not be used as is in automatic summarizers especially when given the fact that the repositories data are to be processed by machines, not humans. Hence, a process I call features extraction and generation has to be applied to the used repositories. The generated features are meant to be in a format usable by the automatic summarizers.

In this chapter I discussed the need for performing the features generation task, what repositories are used and how the extracted features will help with computing the semantic distance between concepts. Details of how the extracted features from WordNet are used with the summarization framework I built is presented in the next chapter. In the following chapter, I describe how the Wikipedia-assisted summarization framework was built, the details of its features generations process and how the summarizer functions.

Chapter 4

Summarization Aided with WordNet

In the previous chapter, I highlighted the need for a suitable repository for machines to better handle single or multi-documents summarization. In this chapter, I describe how WordNet was used for this task and give more details about the implementations and system design. I also report the evaluations that were performed on the system and the results obtained.

4.1 Overview

Figure 4.1 illustrates the architecture of the system and the major tools used to complete the summarization process. It can be noted a server/client model has been adopted during the implementation of the system. The user makes a request that gets passed to the system core. The request can be the documents to be summarized in addition to a query if desired. The documents and the query are preprocessed first, then analyzed and synthesized before the result, which is the summary, is generated. External tools have been used to aid in several tasks during the preprocessing and analysis stages namely, The General Architecture for Text Engineering (GATE) and the Pure Java WordNet Similarity Library (PJWSL). Detailed explanation about summarization process and its stages is given in the following sections. However, it maybe suitable to give a brief overview first on the major external tools that were used during the implementation of the system.

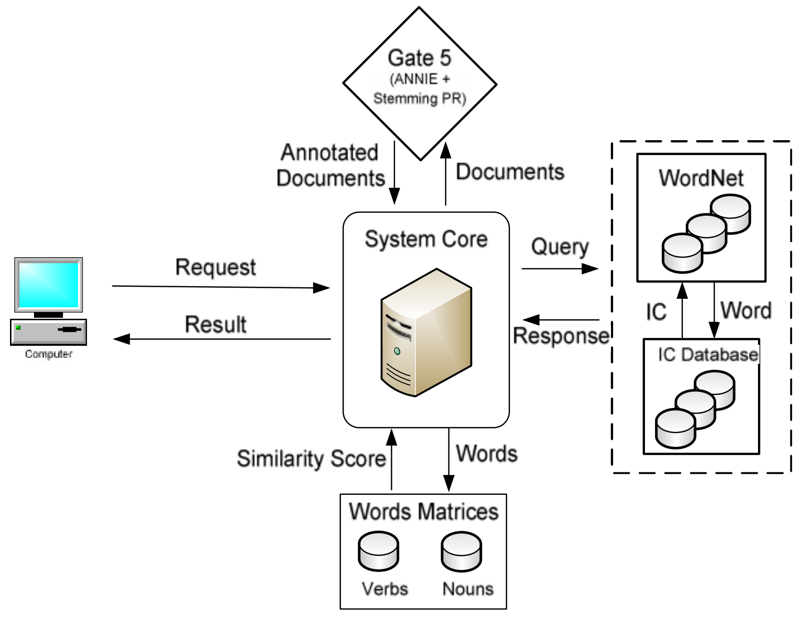


Figure 4.1: Architecture of the WordNet-Aided Summarization system

GATE

GATE is an infrastructure popular in the NLP field and is used for the development and deployment of components that process human languages. It was developed in 1996 at the University of Sheffield. Among the main factors that contributed to its creation are enabling the collaboration and reuse of components, comparing and evaluating them in different tasks, and ensuring the robustness and efficiency of NLP-related systems [143].

It comes with a set of essential tools which are useful for the development of Natural language systems. Such tools include a parser, tokenizer, gazetteer, POS tagger and NE Recognizer. GATE is composed of primarily three subsystems [144]:

- Gate Database Manager (GDM) is a database for storing lexicons, corpora, documents and text in general. It is a hub that all communication of other components goes through.
- Gate Graphical Interface (GGI) is a tool used to view and access the services provided by the GATE components. It can be used to create documents, which are

stored in GDM, view and edit them, and display the results of components executions. Figure 4.2 shows an example GGI.

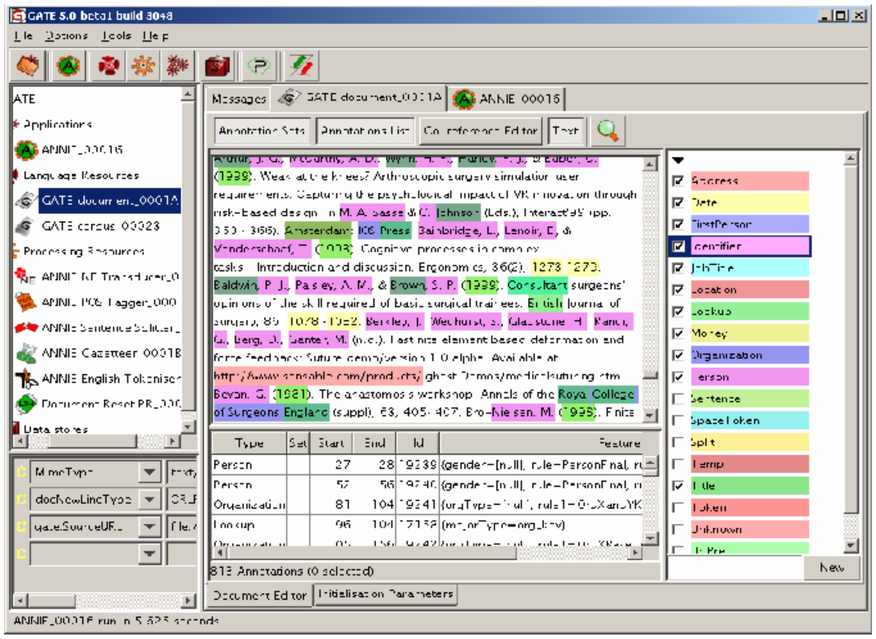


Figure 4.2: GATE Graphical Interface (GGI)

- Collection of REusable Objects for Language Engineering (CREOLE) is a set of Language Engineering (LE) modules that performs the analysis work and discovers information within any fed document. A CREOLE component can also be a wrapper for a set of other modules such as parsers, tokenizers and POS taggers.

GATE also comes with an Information Extraction component called ANNIE (A Nearly New IE). It is designed to be flexible and usable in many different applications handling different types of text documents for different purposes [145]. It consists of the following modules: tokenizer, sentences splitter, POS tagger, gazetteer, finite state transducer, orthomatcher, and coreference resolution. In my work, I use the following modules of ANNIE in the preprocessing stage: tokenizer, sentences splitter and POS tagger. In addition, GATE is used to help parse different types of documents with different formats. Custom versions of GATE and ANNIE were created as java library classes and embedded in my system.

PJWSL

The PJWSL library was developed by Ted Pedersen as a pure Java alternative to the popular WordNet::Similarity [146] Perl library. It also contains an implementation of the JCn [47] and Lin [48] semantic similarity measures in pure Java. To interact with WordNet, access its contents and have access to JCn's and Lin's semantic similarity measures, the PJWSL [147] library was integrated in my system in a pre-processing step. The library was used to build two matrices containing the semantic similarity score between all nouns and all verbs in WordNet by employing the JCn similarity measure.

The system described in this chapter advocates a trade-off methodology between extractive and query-based summarization and is aided with an experts-made repository. The former is due to the fact that the developed methodology uses a scoring function which employs WordNet taxonomy to generate sentence-sentence semantic similarity as well as a set of features to quantify the relevance of each sentence. This yields a summary formed by the highest-ranked sentences. On the other hand, it is also query-based due to the explicit accounting of the topic-sentence semantic similarity in the overall methodology as detailed in the next sections.

My system assigns a score to each sentence in the source documents based on a set of static and dynamic features. Static features include sentence's locations and the number of Named Entities (NEs) in each sentence. Dynamic features on the other hand are those that change based on the document sets chosen. The score given for the similarity between a sentence, and the rest of the sentences in the documents set is an example of a dynamic feature employed in my system.

WordNet, the repository that aids the system, organizes nouns and verbs into hierarchies of “IS-A” relations. While WordNet includes adjectives and adverbs, they are not organized into “IS-A” hierarchies, and so similarity measures can not be directly applied. In an attempt to expand the process of measuring the semantic similarity between sentences, a novel approach is suggested and implemented to utilize adjectives and adverbs by exploiting the inferred relationships from words attributes in WordNet. In effect, adjectives and adverbs are eventually transformed to nouns/verbs and employed in the semantic relatedness computing process.

4.2 System Stages

As mentioned in the previous chapter, there are three major stages involved in summarization: Parsing and Preprocessing the source documents, Analyzing the documents and their Features, and synthesizing the summaries. Figure 4.3 shows the major stages and their subcomponents as implemented in my system. The summarizer accepts as input: the documents to be summarized and optional parameters supplied by the user. The parameters include: redundancy threshold, summary limit, weight given to title/query, and the sentences similarity threshold. Each of these parameters is explained in their corresponding modules in the processing and post-processing stages.

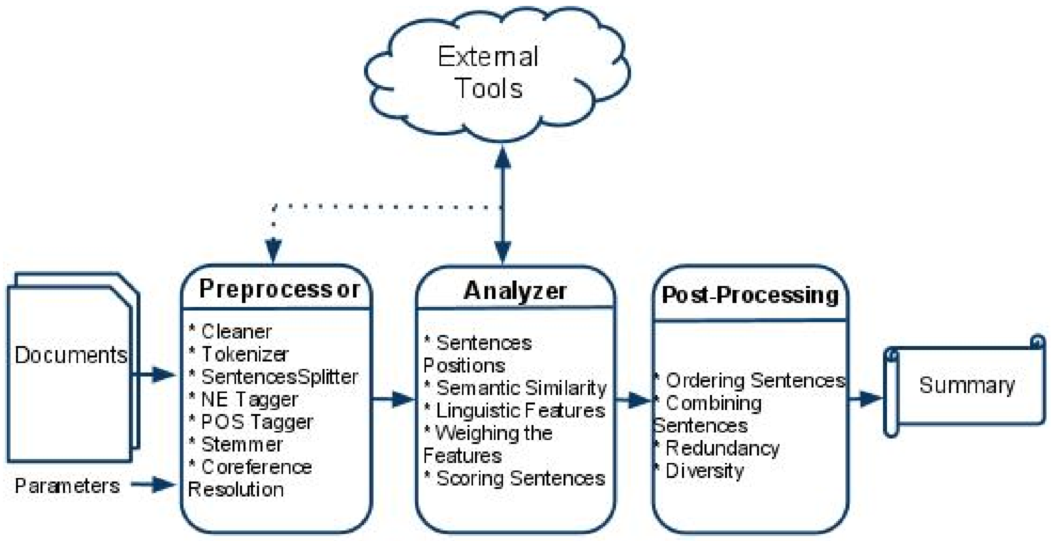


Figure 4.3: Stages of the WordNet-backed Summarizer

Figure 4.4 shows a UML class diagram[10] illustrating the main classes in the implemented summarization module. The classes *DocumentSet*, *Document*, *Sentence*, *Word* and *NamedEntity* are self-explanatory and represent the document sets, documents, sentences, words and detected Named Entities, respectively. *GATEBroker* is used to establish a connection with GATE's ANNIE resources which are employed by the classes *textProcessor* and *documentPreprocessor* to help preprocess documents and tag them with useful features for subsequent processing as will be described in the following sections. The class *WordNetBroker* is used for managing a connection with WordNet and the retrieval of relevant data as needed in the processing and post-processing stages. It is also used in conjunction with the classes *SentencesRanker* and *SimilarityMeasure* to score sentences based on the chosen similarity measure in the algorithm implemented.

[10] This is a simplified class diagram and contains only the main segments of the total class information. Much of the details have been omitted to focus on most important and relevant features to this thesis.

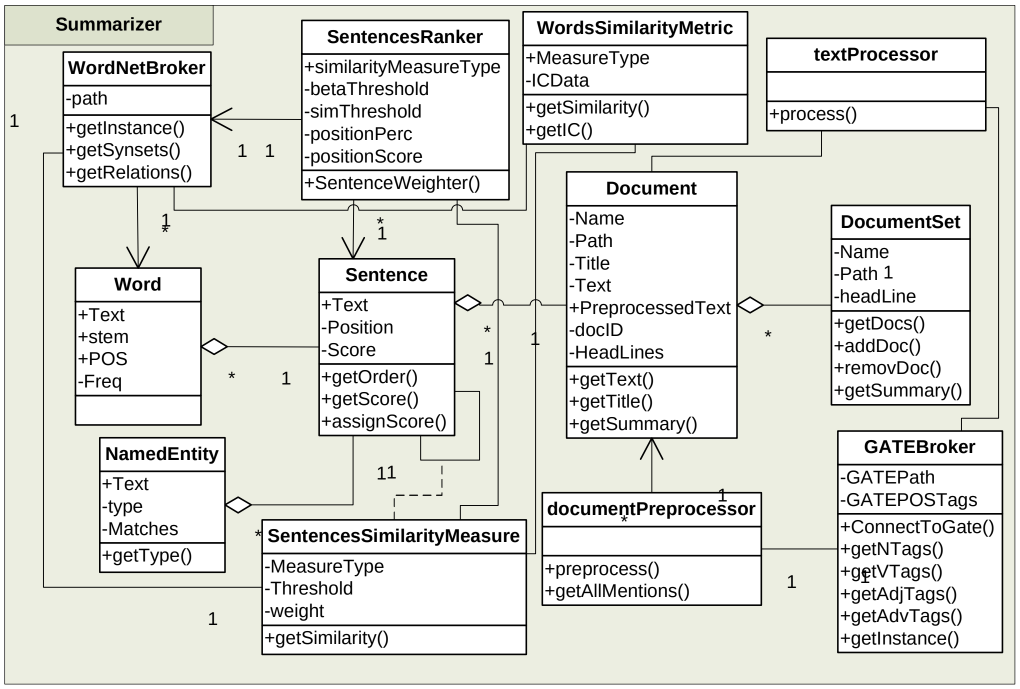


Figure 4.4: Class diagram of the *WordNet-based summarizer* package

The following sections give an overview on the main subcomponents involved in each stage and highlight the additions made to improve the system.

4.2.1 Preprocessing

The preprocessing stage involves cleaning the source documents, splitting, annotating and tokenizing the sentences, and extracting the features. The preprocessing tasks are handled by the classes *documentPreprocessor* and *textProcessor* and their associated models as illustrated in Figure 4.5 and Figure 4.6.

Cleaning: First, the format of the document is detected and handled by the model *documentPreprocessr* shown in Figure 4.5. Then, unnecessary information and tags are removed from the source documents such as the HTML/XML mark-up tags, news agencies names appearing at the beginning of documents and tables containing numbers. Afterwards, key parts from the documents are extracted such as the publication dates, the documents IDs, and the headlines/titles if exist in the documents. The document ID and publication date along with the document name are used to identify each document during the different processing stages.

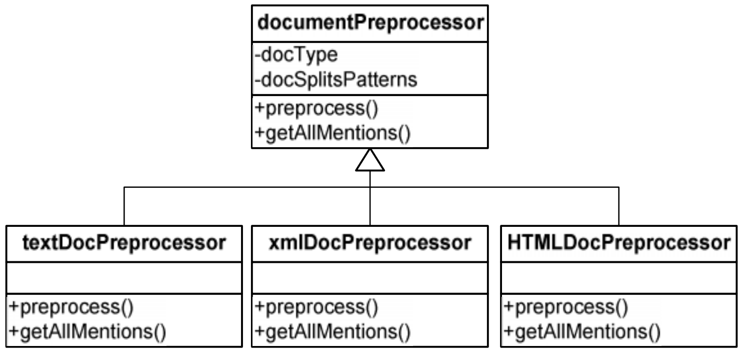


Figure 4.5: Class Diagram of the *documentPreprocessor* model in the WordNet-based Summarizer

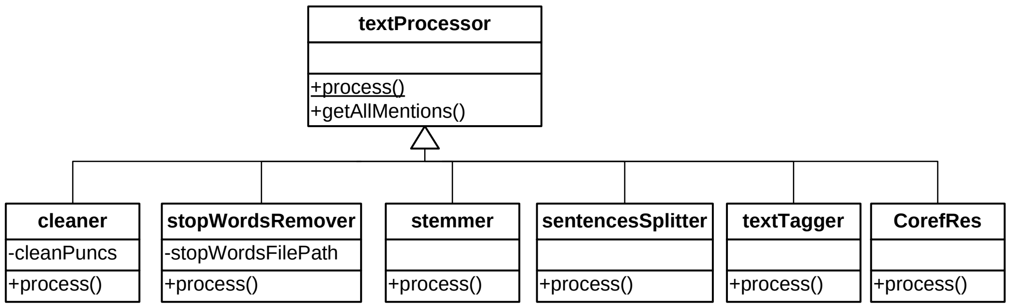


Figure 4.6: Class diagram of the *textProcessor* model in the WordNet-based summarizer

Tokenizer and Sentences Splitter: Sentences and word boundaries are then detected and different features are extracted with the help of GATE from the source sentences and the provided user query. The splitter uses a gazetteer of abbreviations to help distinguish sentence-marking full stops from other kinds. The splitter is also enhanced with RegEx-based rules that improve the execution time and robustness of the system. The system uses custom version of GATE ANNIE for achieving sentences splitting in which it also tries to resolve some of the scenarios not handled well with GATE ANNIE splitter such as not allowing sentences to start with numbers.

POS and NE Tagger: tokenized words are annotated with POS tags which are used in subsequent stages. Finding the Named Entities mentioned in text such as Locations, Organizations and Persons is also performed at this stage. ANNIE is used for both POS and NE tagging.

Stemming: While tokenizing the documents, a "stem" feature is applied to every token with the word stem as its value. A Processing Resource (PR) in GATE is used to perform the stemming by applying the Porter Stemmer Algorithm [148].

Coreference Resolution: As for co reference resolution, three GATE modules are used: *Orthomatcher*, *Pronominal Coreferencer*, and *Nominal Corerferencer*. *Orthomatcher* uses a lookup table of aliases that stores non-matching strings representing the same entity: e.g. "Coca-Cola" and "Coke". The *Pronominal Coreferencer* module performs anaphora resolution using the JAPE grammar rules. For each pronoun, the coreference module generates an annotation of type "Coreferences" containing two features: the antecedent offset, and matches. During the preprocessing stage, each entity is replaced with its longest alternative "match", if it's not found to be the longest. For example, the sentences:

- Barak Obama was born in Hawaii.
- Mr. Obama was elected to the Illinois Senate in 1996.
- He was re elected to the Illinois Senate again in 1998.

In the above three sentences, the longest alternative to "Mr. Obama" and "He" is "Barak Obama", and thus the sentences are rewritten as follows:

- Barak Obama was born in Hawaii.
- Barak Obama was elected to the Illinois Senate in 1996.
- Barak Obama was re elected to the Illinois Senate again in 1998.

Since the used modules were designed to be applied on single documents, cross-document coreference resolution was not achievable. When dealing with multi-documents datasets, the Coreference resolution module is applied to the documents individually in the pre-processing stage before moving on to the following stages.

4.2.2 Analysis

After preprocessing the documents and queries, the processing stage begins scoring sentences based on the computed/extracted set of features. I discuss the chosen features first, and then I describe how the similarity between sentences and text fragments is computed.

4.2.2.1 Summarization Features

Each sentence is given a score implying its significance in relation with the rest of the sentences. The scores are the results of linear combination of the weights given to each feature. The features taken into account during the Analysis stage as follows:

Sentences Location

The position of the sentences in the document can play a significant factor in finding the sentences that are most related to the topic of the document [149]. Therefore, the position of sentences was taken into account when computing a score for each sentence. A score of 1 is given to the first and last sentences in the document. The rest of the document sentences were given equally 0.5.

Named Entities (NE)

In a study conducted in [150], it was found that 73-87% of all web queries contain Named Entities. This finding demonstrates that a high percentage of all web queries recognizably target entities. The system developed here is a query-based summarizer. Thus, it was important to conduct an analysis on the NEs in the processed documents, and the queries. The system uses ANNIE to recognize NEs (Locations, Organizations, and Persons) and relate them with their references.

Title / Query

The evaluated similarity of each sentence on one hand, and title and/or query on the other hand is taken into account when scoring sentences. All the preprocessing stages are applied to both the query and title. If many documents are passed to the system at a time and a generic "Topic" is provided for the fed documents, the "Topic" is treated just as if it were a user's query in the algorithm described next. Documents titles are usually embedded at the beginning of the source document, and are extracted and identified at the preprocessing stage.

4.2.2.2 Sentence-Sentence Similarity

I define here several Sentences Similarity measures implemented in the built summarization framework. These measures are illustrated in the classes generalized by the *SentencesSimilarityMeasure* model in Figure 4.7. The *WordNetBroker* class provides an interface to WordNet and allow for retrieving its synsets and relations. *WordsSimilarityMetric* gives access to the implementation of JCn's and Lin's words similarity metrics. JCn's was implemented as the default IC metric in the framework, with an option to switch to Lin's if needed. Figure 4.8 shows the class diagram for JCn's and Lin's metrics. In the following subsections I give an overview on each of the implemented sentences similarity measures.

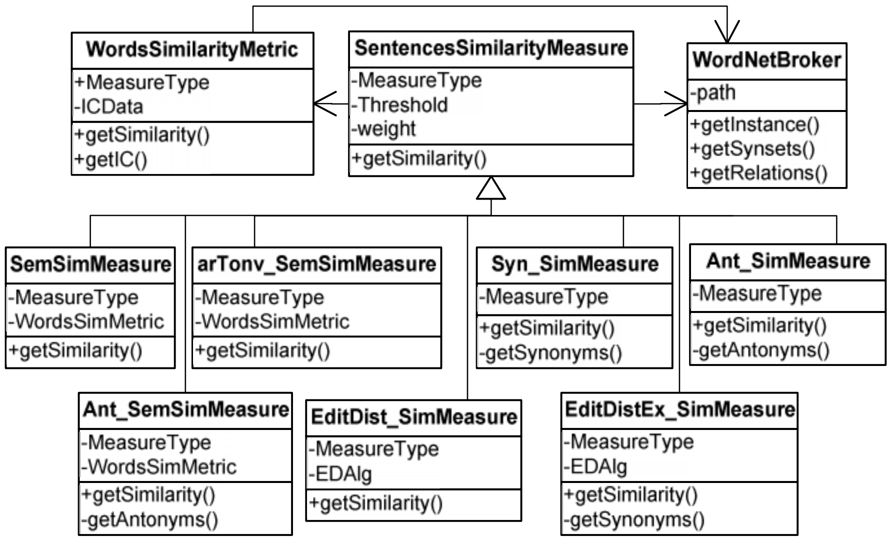


Figure 4.7: Class diagram of the *SentencesSimilarityMeasure* model in the WordNet-based summarizer

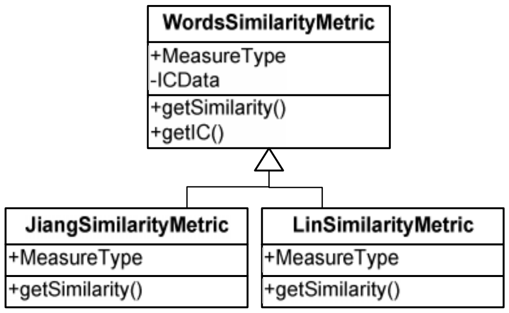


Figure 4.8: Class Diagram for JCn's and Lin's words similarity metrics

A- Sentences Semantic Similarity Measure (SemSimMeasure)

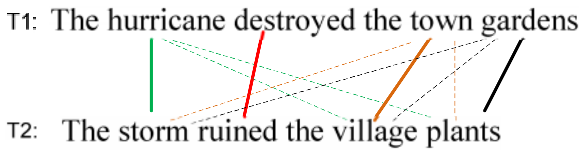
To compute the semantic similarity between any two sentences, this measure considers only the explicitly mentioned nouns and verbs in each sentence. It works by computing the semantic similarity between nouns from the first sentence with nouns from the second and verbs from the first sentence with verbs from the second. The following formula is used for finding the similarity:

$$Sim(Sent1, Sent2) = \frac{Sim_w(WordN1, WordN2) + Sim_w(WordV1, WordV2)}{PairsCounter} \qquad \mathbf{4.1}$$

Where:

- WordN1 and WordN2 are the nouns taken from sentence1 and sentence2 respectively.
- WordV1 and WordV2 are the verbs taken from sentence1 and sentence2 respectively.
- PairsCounter is the total number of word pairs chosen from the two sentences.

Following is an example illustrating this with two sentences: T1 and T2. The nouns from the first sentence are compared with all the nouns from the second while verbs in the first are compared with verbs from the second. For noun or verb from the first sentence, I compare its similarity with all of the nouns/verbs in the second sentence and choose the highest similarity score for each pair. In the given example, the noun *hurricane* gets its highest similarity score with the term *storm*, while *town* is paired best with *village*. The verb *destroyed* is paired with the only verb in the second sentence, *ruined*.


T1: The hurricane destroyed the town gardens
T2: The storm ruined the village plants

As for computing the similarity between words in Sim_w, two semantic metrics were considered, namely Lin's [48] and JCn's [47] with the default being the latter.

B- Transforming Adjective/Adverbs to Nouns/Verbs (arTonv_SemSimMeasure)

In most of the implemented summarization systems that use WordNet-based semantic similarity measures, the semantic similarity between sentences rely heavily on the nouns/verbs existing in each sentence. This is mainly due to the way WordNet was designed. In WordNet, IS-A relations are defined for only nouns and verbs. They also do not cross part-of-speech boundaries. In other words, nouns can be semantically compared with only nouns, and verbs with verbs. At first, a basic analysis was performed on adjectives/adverbs by using descriptors and expanding adjectives/adverbs with their synonyms. However, this approach can provide counterintuitive results for some sentences as it lacks a deep analysis on adjectives/adverbs that might have corresponding nouns/verbs in other similar sentences. The following sentences provide an example for the mentioned problem, with key adjectives/adverbs in Italic style:

1. A *careful* examination of the warehouse *dangerous* tools occurred yesterday.
2. The warehouse *rusty* tools were examined yesterday.
3. The hazardousness of the tools in the warehouse was evaluated yesterday.

A basic analysis based on only synonyms-expansion which extracts nouns/verbs from each sentence indicates that sentences 1 and 2 are more similar than sentences 1 and 3. This is mainly due to the lack of analysis for the adjective "*dangerous*" (which is related to "*hazardousness*") and the adjective "*rusty*" in sentence 2. Intuitively, sentences 1 and 3 are more related and similar than sentences 1 and 2. To overcome this problem, a deeper analysis has to be applied on the incurred adjectives and adverbs.

In an attempt to overcome the mentioned problem, the summarizer aims to extract implicit knowledge from WordNet and utilize it to transform adjectives/adverbs to corresponding nouns/verbs. This is achieved by taking advantage of the additional non-hierarchical relations appearing in WordNet's adjectives/adverbs synsets.

In WordNet 3, every synset is connected to other synsets via a number of relations. These relations vary based on the type of the word. Hypernyms and hyponyms only exist for nouns and verbs. The attempt made here is to stretch WordNet by utilizing its words attributes for adjectives and adverbs and substitute their lack of hyponyms by transforming them to their corresponding nouns or verbs. *Pertainym* relations, *derivational* links, *synonyms*, and *root adjectives* are the attributes used for this purpose. Not all attributes are defined for every adjective and adverb in WordNet. It is only necessary for some of them to be found to get a corresponding noun/verb to the targeted adjective/adverb. The function written for this task chooses the first encountered noun representing the adjective (and verb if found for adverbs). For example, the adjective "American" has two senses in WordNet and both have "America" as the noun referred to by the derivational links. The adverb "definitely" has a *pertainym* relation with the adjectives "decided", "unquestionable" and "emphatic". The two adjectives "unquestionable" and "emphatic" are *derivationally* related to the nouns: "unquestionableness", "unquestionability" and "emphasis". The pseudo code of the implemented function is shown in Figure 4.9 where "word" is the passed adjective/adverb to the function.

After transforming adjectives/adverbs to their corresponding nouns/verbs using the above described function, Formula 4.1 is used to compute the similarity between two sentences. The process for computing the scores between sentences 1 and 2 is also illustrated in Figure 4.10.

The effect of this is to get a score which takes into account every noun, verb, adjective, and adverb in both sentences. Expression 4.1 is also used to determine the score attached to the semantic similarity of the sentence to the query and the title, if they exist. For example, in two sentences such as:

T1: The hazardousness of the tools in the warehouse was evaluated yesterday

T2: A *careful* examination of the warehouse *dangerous* tools occurred yesterday

```
Procedure getNV(word)
       if (word ∈ nounsList ) OR (word ∈ verbsList)
              return word
       rtn ← GetDeriWord(word)
       if (rtn is NOT empty){
              return rtn
       }

# if still not found, get the word “valid forms” (i.e. shaved => shave, eaten =>eat)
# (alternate spellings, conjugations, plural/singular forms, etc.)

       ValidFormsN_V,ValidFormsAdv,ValidFormsAdj ← getValidForms(word)
       for each (VF ∈ ValidFormsN_V){
              if (VF ∈ nounsList ) OR (VF ∈ verbsList)
                     return VF
       }
       for each (VF ∈ ValidFormsAdj){
              if ( VF ∈ adjsList)
                 Sims ← getSimilarTo(VF) // using the “similar to” relation
                     for each (sim ∈ Sims){
                       rtn ← GetDeriWord(sim)
                       if (rtn is NOT empty){
                                    return rtn
                       }
                     }
                 Attrs ← getAttr(VF) // using the “Attribute” relation
                     for each (atr ∈ Attrs){
                       rtn ← GetDeriWord(atr)
                       if (rtn is NOT empty){
                                    return rtn
                       }
                     }
              }

       }
       for each (VF ∈ ValidFormsAdv){
              if ( VF ∈ advList)
                 Sims ← getSimilarTo(VF) // using the “similar to” relation
                     for each (sim ∈ Sims){
                       rtn ← GetDeriWord(sim)
                       if (rtn is NOT empty){
                                    return rtn
                  }
                     }
                 Perts ← getPert(VF) // using the “Pertainyms” relation
                     for each (Prt ∈ Perts){
                       rtn ← GetDeriWord(Prt)
                       if (rtn is NOT empty){
                                    return rtn
                  }
                     }

                 Attrs ← getAttr(VF) // using the “Attribute” relation
                     for each (atr ∈ Attrs){
                       rtn ← GetDeriWord(atr)
                       if (rtn is NOT empty){
                                    return rtn
                  }
                     }
              }
       return word
}
Procedure GetDeriWord(word) {
              Senses ← Get_all_senses(word)
              for each (sense ∈ Senses){
                     deri ← Get_derivation_link(sense)
                     if (deri is NOT empty){
                          return getFirstItem(deri)
              }
}
```

Figure 4.9: Pseudo code of the implemented function for converting adv/adj to nouns/verbs

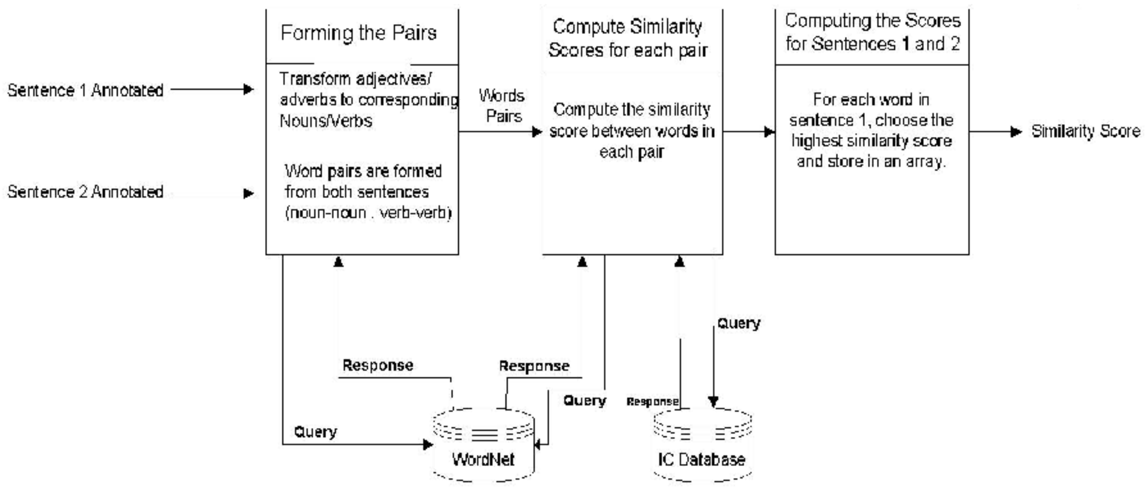

Forming the Pairs
Transform adjectives/ adverbs to corresponding Nouns/Verbs
Word pairs are formed from both sentences (noun-noun . verb-verb)
Sentence 1 Annotated
Sentence 2 Annotated
Words Pairs
Compute Similarity Scores for each pair
Compute the similarity score between words in each pair
Computing the Scores for Sentences 1 and 2
For each word in sentence 1, choose the highest similarity score and store in an array.
Similarity Score
Query
Response
Response
Query
Response
Query
WordNet
IC Database

Figure 4.10: Computing the similarity score between sentences 1 and 2 with the aid of WordNet

The adjectives *careful* and *dangerous* are transformed to *carefulness* and *danger*, respectively. When computing the similarity between T1 and T2, the following pairs are considered:

> (hazardousness, danger), (tools, tools), (warehouse, warehouse), (evaluated, occurred), (yesterday,yesterday)

Where first term of each pair is taken from T1 while second term is taken from T2.

C- Expanding words with Synonyms (Syn_SimMeasure)

For this measure, the terms are expanded with their synonyms using WordNet first. This is followed by a simple words-matching process that takes place between the tokens of each pair of sentences. If a word in the first sentence is found to match a word (or its synonyms) in the second, it is a *hit*. For example, the sentence "*Harry has a grin*" would be expanded to "*Harry grin smile*". The expanded sentence is then compared with the terms in the second sentence and if there is a match, I have a *hit*. The average is taken by dividing the total number of hits by the number of word-pairs compared between the two sentences.

Assume that there are two sentences ***A*** and ***B*** where the length of ***A*** is ***m*** while the length of ***B*** is ***n***. Each word in sentence ***A*** is accessed by ***i*** while words in sentence ***B*** are accessed by ***j***. This also means that the maximum value ***i*** would have is ***m*** while the maximum value for ***j*** is ***n***. For example, when referring to the second word in sentence ***B***, I use ***B[2]***. To show how to compute the similarity between ***A*** and ***B*** using this measure, I provide the pseudo code of the implemented algorithm in Figure 4.11.

```
Hits ← 0

MatchFunctionStart
    ForEach A[i] in A
        ForEach B[j] in B
            If B[j] is not Matched and A[i]  belongs to Syn(B[j])
                Mark B[j] as a match for A[i]
                Hits ← Hits+1
                Break
            EndIf
        EndForEach
    EndForEach
MatchFunctionEnd

SimScore ← Hits/(PairsCounter)
```

Figure 4.11: Pseudo code of the algorithm used for computing the similarity between words based on the Syn_SimMeasure

For example, in two sentences such as:

T1: The student bought a book.

T2: The pupil purchased a book.

I have the terms *pupil, purchased* and *book* from the second sentence expanded as follows:

(student , pupil, educatee, scholar, scholarly person , bookman), (buy, purchase, bribe, corrupt, grease one's palm) , (book, volume, ledger, account book, book of account, record, record book, script, playscript, rule book, Koran, Quran, Bible, Christian Bible, Good Book Holy Scripture, Holy Writ, Word of God, Word)

When the similarity between the two sentences is computed, the following pairs are considered:

(Student,pupil), (bought, purchased), (book,book)

Where first term of each pair is taken from T1 while second term is taken from T2.

D- Replacing words with Antonyms (Ant_SimMeasure)

For this measure, I expand the words in the second sentence by replacing them with their antonyms using WordNet's relations. I then compare the original words in the first sentence with the antonyms in the second. If there is a match, it is a *hit*. The average is then taken by dividing the number of hits by the total number of word-pairs. In effect, this measure is used to show how dissimilar or *diverse* two sentences are. Assuming that this measure needs to be applied to two sentences ***A*** and ***B***, I apply the pseudo code shown in Figure 4.12. The definitions of the terms ***i***, ***j***, ***A[i]*** and ***B[j]*** in the figure are the same as those given for the previous measure. The function *Ant(X)* generates a list of antonyms for the word X.

```
Hits ← 0

MatchFunctionStart
        ForEach A[i] in A
            ForEach B[j] in B
                If B[j] is not Matched and A[i]  belongs to Ant(B[j])
                    Mark B[j] as a match for A[i]
                    Hits ← Hits+1
                    Break
                EndIf
            EndForEach
        EndForEach
MatchFunctionEnd

SimScore ← Hits/(PairsCounter)
```

Figure 4.12: Pseudo code of the algorithm used for applying the measure Ant_SimMeasure on two Sentences A and B.

Antonym word pairs carrying contraries meaning as in “soft-hard” and “large-small” can implicitly carry different degrees of oppositions as in “so large” or “very soft”. Contradictions on the other hand do not carry different degrees of oppositions as in the pairs “left-right” or “empty-full”. In the built and implemented model, I try to capture all types of antonyms regardless of the embedded differences in the antonym logical relations. The results of applying the diversity metric on all types of antonym relations can lead to the detection of sentences carrying different sentiments, different information, contraries or contradictions as shown in the following examples.

Different sentiments:

The team is *happy* to be back.

The other team is *sad* for losing the match.

Different information

Iraq finally declared *war* and invaded Kuwait.

Peace treaty was announced between Kuwait and Iraq.

Contraries and contradictions:

The store was found to be *empty* when the police arrived.

The police reported that the store was *full* of counterfeit products.

For example, in two sentences such as:

T1:The team is happy for winning the match.
T2: The other team is *sad* for *losing* the match.

The antonyms generated for the terms in the second sentence are:

(Glad, joyful, good, happy, felicitous), (keep, win, find, regain, profit, hold on, break even, acquire, gain)

When the diversity between the two sentences is computed, the following pairs are considered:

(happy, sad) , (win, lose)

Where first term of each pair is taken from T1 while second term is taken from T2.

E- Replacing words with Antonyms and computing SemSimilarity (Ant_SemSimMeasure)

Just as was performed with the measure *Ant_SimMeasure*, I replace words in the second sentence with their antonyms using the different synsets relations in WordNet. Instead of performing simple words matching, I compute the semantic similarity between sentences words using the JCn's metric. Assuming that this measure needs to be applied on two sentences ***A*** and ***B***, I apply the pseudo code shown in Figure 4.13. The definitions of the terms ***i***, ***j***, ***A[i]*** and ***B[j]*** in the figure are the same as those given for the above measure. SemSim refers here to the semantic similarity between terms using the JCn metric.

```
Hits ←0

MatchFunctionStart
        ForEach A[i] in A
            ForEach B[j] in B
                If B[j] is not Matched and SemSim(A[i], Ant(B[j]))>Threshold
                    Mark B[j] as a match for A[i]
                    Hits ← Hits+1
                    Break
                EndIf
            EndForEach
        EndForEach
MatchFunctionEnd

SimScore ← Hits/(PairsCounter)
```

Figure 4.13: Pseudo code of the algorithm used for applying the measure Ant_SemSimMeasure on two Sentences A and B.

Again, as with *Ant_SimMeasure*, this measure is most suitable for computing the diversity between sentences. For example, in two sentences such as:

T1: The new car is a delight to John.

T2: Leaving the car was a *grief* to John.

The antonyms generated for terms in the second sentence is

Joy

When the diversity between the two sentences is computed, the following pairs are considered:

(delight, joy)

Where first term of each pair is taken from T1 while second term is taken from T2.

F- Edit Distance (EditDist_SimMeasure)

For this measure, I use the Levenshtein metric [151] to compute the edit distance between every pair of sentences. The edit distance is defined as the minimum number of operations required to transform one text to another using insertion, deletion or substitution operations on single characters. Assuming that this measure needs to be applied on two sentences ***A*** and ***B***, I apply the pseudo code shown in Figure 4.14. The definitions of the terms ***i***, ***j***, ***A[i]*** and ***B[j]*** in the figure are the same as those given for the above measure.

```
Hits ← 0

MatchFunctionStart
    ForEach A[i] in A
        ForEach B[j] in B
            If B[j] is not Matched and LDSim(A[i], B[j])>Threshold
                Mark B[j] as a match for A[i]
                Hits ← Hits+1
                Break
            EndIf
        EndForEach
    EndForEach
MatchFunctionEnd

SimScore ← Hits/(PairsCounter)
```

Figure 4.14: Pseudo code of the algorithm used for applying the measure EditDist_SimMeasure on two Sentences A and B.

To get the value of *LDSim*, I apply the following expression:

$$LDSim(w_1, w_2) = 1 - \frac{EditDist(w_1, w_2)}{MaxLen(w_1, w_2)} \qquad \textbf{4.2}$$

For example, LDSim(XYZ,WDT) would give a 0 while LDSim(XYZ,XDFM) results in 0.25. This measure helps in detecting the largest common sub sequences among sentences and assigns weights based on their detection. In addition, it can help in weighing Named Entities which are not necessarily detected by the external Named Entity recognizer used in other parts of the summarization stage. For example, in two sentences such as:

T1: John lived in apartment3.
T2: John was found dead in apartment4.

The similarity between the two sentences is computed by considering the following pairs:

(John, John), (apartment3, apartment4), (lived, found)

Where first term of each pair is taken from T1 while second term is taken from T2.

G- Edit Distance with Synonyms Expansion (EditDistEx_SimMeasure)

Similar to *EditDist_SimMeasure*, this measure utilizes the Levenshtein metric for computing the edit distance between sentences. However, for every sentences pair it differs by expanding the words of the second sentence with their synonyms to provide a semantic-based metric. When two words are compared, I expand the second word with its synonyms and compute the edit distance between all words pairs including the synonyms. Assuming that this measure needs to be applied on two sentences ***A*** and ***B***, I apply the pseudo code shown in Figure 4.15. The definitions of the terms ***i***, ***j***, ***A[i]*** and ***B[j]*** in the figure are the same as those given for the above measure.

```
Hits ← 0
MatchFunctionStart
    ForEach A[i] in A
        ForEach B[j] in B
            If B[j] is not Matched and LCDSim(A[i], syn(B[j]))>Threshold
                Mark B[j] as a match for A[i]
                Hits ← Hits+1
                Break
            EndIf
        EndForEach
    EndForEach
MatchFunctionEnd
SimScore ← Hits/(PairsCounter)
```

Figure 4.15: Pseudo code of the algorithm used for applying the measure EditDistEx_SimMeasure on two Sentences A and B.

As in the previous variation, the *syn()* function refers to obtaining the synonyms for the given word using WordNet. For example, in two sentences such as:

T1: John was discovered dead in apartment2

T2: John was found dead in a flat

An expansion of the terms in the second sentence takes place by finding their synonyms as follows:

(find, happen, chance, bump, encounter, detect, observe, discover, gain, notice, regain, determine, ascertain, feel, see, witness, receive, get, rule, recover, retrieve), (dead, asleep, assassinated, bloodless, cold, d. o. a., deathlike, defunct, doomed, executed, fallen, lifeless, murdered, nonviable, slain, extinct, barren, non-living), (flat, apartment, plain, field, box, freight car, pneumatic tire, pneumatic tyre, scenery, scene, housing, lodging, living accommodation)

The similarity between the two sentences is computed by considering the following pairs after considering all synonyms:

(John, John), (discovered, Syn(found)), (dead, dead), (apartment2, syn(flat))

Where first term of each pair is taken from T1 while second term is taken from T2.

4.2.2.3 Scoring the Sentences

Each sentence is given a score representing its importance. The score for each sentence, (i), is simply the linear combination of the weights given for each feature. The highest scoring sentences are selected as candidates to be part of the summary in the last stage. The number of sentences displayed is controlled by the user according to the summary limit threshold. The formula used for assigning a score to each sentence is:

$$\text{Score(i)} = \frac{(\alpha Sim(s_i, \mathrm{T}) + (1-\alpha) Sim(s_i, Q))\ n(s_i)\ (F_{NE}(s_i) + 1)\ P(s_i)}{N(NE+1)} \qquad \textbf{4.3}$$

Where:

- N = the total number of sentences
- $n(s_i)$ = The number of sentences that have similarity score bigger than a pre-defined threshold value
- P(s) = either 1 for sentences appearing at the top and end of the document, or 0.5 for the rest.
- $Sim(s_i, T)$ and $Sim(s_i, Q)$ are for the Similarity between the Title and the Query, respectively, and the sentence s_i
- $F_{NE}(s_i)$ = the number of Named Entities contained in the sentence s_i
- NE: the number of Named Entities in the document.

The rationale behind the preceding is to allow the score assigned to the sentence i to be very much dependent on the evaluation of the similarity of i to both the title and the query using a combination of both entities. This output is weighted by $n(s_i)$, which expresses, to some extent, the frequency of the sentences in the document(s) that are similar to i up to some threshold, as well as the number of Named Entities in the sentence and its position. The positioning parameter is motivated by the observation that usually, the beginning and end of documents contain more significant information regarding the context of the underlying document(s) as authors attempt to provide concise overview at the beginning and concluding remarks at the end. But, obviously this is very much context dependent. The weighting parameter α is left open to the choice of the user depending on his/her prior knowledge about the relevance of the title and/or query. In the absence of any further evidence, the default value is 0.5, which is in agreement with the principle of indifference in statistics.

4.2.2.4 Generating Summaries

After scoring the sentences, the synthesizer is fed with the highest scoring candidate sentences in order to generate and produce the summary in the desired format. A basic redundancy reduction module is implemented and is described along with the other modules in the next few subsections.

Combining Sentences

Multi-document summaries are generated in a similar fashion to the single-document summaries by computing sentences scores in each document separately and then choosing the highest scoring sentences from all documents.

Redundancy Reduction

To reduce the redundancy between the highest scoring sentences, a redundancy threshold is defined and can be passed by the user as a parameter. After adding the first highest scoring sentence to the summary, only sentences that have a similarity score (with the highest scoring sentence) less than the set threshold are added. The process is repeated after adding any high scoring sentence to the summary. In effect, no sentences pair included in the final summary has a similarity score greater than the redundancy threshold. The pseudo code for this process is shown in Figure 4.16.

```
Add highest-scoring sentence to SummarySentences
Remove added sentence from the highest scoring sentences pool
      While words limit not reached and sentences list not exhausted
            Find next highest scoring sentence with score >= zero
            Foreach (Sent in SummarySentences){
                  If ( similarity(Sent, foundSentence) >redundancy_limit){
                           Remove foundSentence from highest-scoring sentences list
                           foundSentence = NULL
                           break
                        }
                  }
            If (foundSentence is NOT NULL){
             If adding the sentence does not increase summary size beyond limit {
                  add the sentence
               }
             }
            Else, remove the sentence from the list
      Iterate through the rest of the sentences
```

Figure 4.16: Psueduocode of the Basic Redundancy Reduction Algorithm in the WordNet-based System

Ordering Sentences

A summary is generated by choosing the most important sentences in a document (or the highest scoring) and arranging them in chronological order (in the same order they appear in, in the original source documents) to ensure the readability of the generated summary.

4.3 Evaluation

To evaluate my system, I participated in the Update Summarization task of Text Analysis Conference (TAC 2008). A maximum of two runs from each participant were accepted and evaluated by the conference organizers. At the time, only the *SemSimMeasure* measure was implemented and thus used by me to submit two separate runs. I report first the results obtained with the submitted runs from the automatic evaluation performed by NIST using ROUGE [119] and BE [152] evaluation metrics, and the manual responsiveness evaluation measure.. Following that, I performed my own evaluation using the same dataset to test the performance of the other developed measures that can aid in computing the similarities between sentences.

4.3.1 Test Data and Metrics

For participating in the TAC 2008 update task, the *SemSimMeasure* measure was adopted when computing the semantic similarity between words. Computing the similarity between words was completed by using JCn's metric. The provided test dataset comprised 48 topics. Each topic had a topic statement and 20 relevant documents which had been divided equally into 2 sets: A and B. Documents in set A always chronologically precedes the documents in set B. The provided test dataset was taken from the AQUAINT-2 collection of news articles[11].

All of the submitted summaries were truncated to 100 words. NIST conducted manual evaluation of summaries contents based on the Pyramid method. Four different NIST assessors would create 100-word reference summaries for each document set that addresses the information need expressed in the topic statement.

Each participant team was requested to submit up to 2 runs ranked by priority (1-2). My team submitted two runs: one (run # 1) in which more weight was given to the topic statement, and the other (run # 34) had more weight given to the headlines. In the scoring formula, α was given a value of 0.75 for run 1, and 0.25 for run 34.

4.3.2 Results

In the update task of TAC 2008, 57 peer summaries were manually evaluated with the Pyramid method, and 71 were evaluated using ROUGE and the Basic Elements evaluation package [13].

Table 4.1 shows the average Recall, Precession and F-measure for the ROUGE-1, ROUGE-2, and ROUGE-SU4 evaluations on the two submitted runs. It can be noted that

[11] The AQUAINT-2 collection is distributed by the Linguistic Data Consortium (LDC) and comprises a subset of the LCD English Gigaword Third Edition. It has approximately 907K text documents spanning the period of 10-2004 to 03-2006. The documents are in English and come from different sources including New York Times, the Associated Press and Xinhua News Agency.

in both runs, the system generally ranked higher in Recall than Precession. This suggests that the system is better at finding relevant content than it is at removing irrelevant content. Also, it can be noted that the run in which more weight was given to the topic statement generally achieved better ROUGE scores than the other run with more weight given to the headlines.

	Run 1			**Run 34**		
ROUGE	**Avg R**	**Avg P**	**Avg. F**	**Avg R**	**Avg P**	**Avg F**
1	0.34463	0.33866	0.34148	0.34022	0.33372	0.33680
2	0.08091	0.07933	0.08008	0.08080	0.07912	0.07991
SU4	0.11852	0.11634	0.11737	0.11706	0.11471	0.11583

Table 4.1: **The ROUGE Scores obtained by my system in the two runs I submitted in TAC08**

Table 4.2 shows the automated evaluations average scores obtained by my submitted runs (with their ranks) in comparison with the 71 peer summaries submitted by the rest of the participants. The scores I obtained were above average for all runs.

Evaluation	**Run (1)**	**Run (34)**	**Best**	**Worst**
ROUGE2-R	0.08091 (25/71)	0.08080 (26/71)	0.10382	0.03343
ROUGESU4-R	0.11858 (23/71)	0.11713 (29/71)	0.13646	0.06517
BE	0.04964 (24/71)	0.04903 (28/71)	0.06462	0.01337

Table 4.2: The automated scores (and ranks) obtained by my system compared with the rest in TAC08

The evaluation in TAC2008 included human judgments of linguistic quality. Table 4.3 shows the results and the rank of my system in respect with the rest in the manual evaluation. The metrics shown in the table are: responsiveness which is how well the summary addresses the user's information need; and linguistic quality. The linguistic quality score is guided by consideration of the following factors:

1. Grammaticality
2. Non-redundancy
3. Referential clarity
4. Focus
5. Structure and Coherence

with the scores being between 1 (very poor) and 5 (very good). The results obtained for the submitted runs were above average as shown in the table. This was expected since the summarizer is extractive and no modifications were made to the sentences. Information redundancy, diversity and coherence are the main factors affecting linguistic quality and overall responsiveness. An attempt is made to address these factors in section 4.4.

	Run (1)	**Run (34)**	**Best**	**Worst**
Avg Linguistic Quality	2.719 (12/58)	2.76 (11/58)	3.073	1.312
Overall Responsiveness	2.427 (15/58)	2.385 (18/58)	2.667	1.198

Table 4.3: Manual Evaluation Results in TAC08

It is interesting to test the impact of using other sentences similarity measures in the built summarization system and evaluate their performances. Ideally, it would be optimal to repeat all of the evaluations performed by TAC08 organizers with different variations of the system using different sentences similarity measures. However, it is a labour intensive task to perform the manual evaluations they performed and is beyond my means. Therefore, I use the ROUGE tool for this task which is widely accepted in the community to provide acceptable evaluation results in comparison with human summaries evaluators.

It is also used by the TAC08 organizers and provides a good mean of reference against the results I obtained from my participation.

After obtaining the official evaluation results from the TAC 08 organizers for the submitted runs, I used the same dataset to evaluate other sentences similarity measures using ROUGE with the same parameters as was used with Run1. Namely, I implemented and evaluated the measures: *SemSimMeasure*, *arTonvSemSimMeasure*, *Syn_SimMeasure*, *EditDist_SimMeasure*, and *EditDistEx_SimMeasure* which were all described in section 4.2.2.2. Because the measures involving the replacement of terms with their antonyms reflect the dissimilarity and diversity between sentences, it was decided to implement these measures for enhancing the overall diversity of sentences in the summary and reducing redundancy as described and evaluated in section 4.4. The method I used for participating in TAC08 was *SemSimMeasure* and it is chosen as the baseline during this evaluation.

The results I obtained are illustrated in Table 4.4. It can be noted that *arTonv_SemSimMeasure* gave the best performance for all ROUGE metrics with the biggest increase over the baseline being for ROUGE1. The next best performing metric in ROUGE1 was found to be *EditDistEx_SimMeasure*. This measure has boosted the performance of the baseline by 2.9% and yielded better unigram matches between the generated summaries and the reference summaries as can be noted from comparisons shown in Figure 4.17. As for ROUGE2 and ROUGESU4, the next best performing measure was found to be the baseline as demonstrated in Figure 4.18 and Figure 4.19.

Evaluation	ROUGE1	ROUGE2	ROUGESU4
SemSimMeasure	0.34463	0.08091	0.11852
arTonv_SemSimMeasure	**0.35801**	**0.08125**	**0.11979**
Syn_SimMeasure	0.33823	0.07984	0.11487
EditDist_SimMeasure	0.34249	0.08011	0.11604
EditDistEx_SimMeasure	0.35357	0.08048	0.11828

Table 4.4: ROUGE evaluation results for the different sentences similarity measures

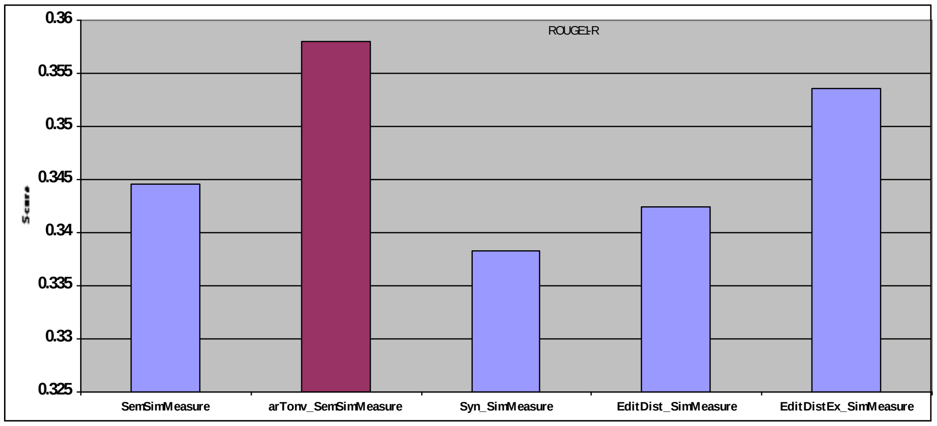


Figure 4.17: ROUGE1 scores showing the performance for the different sentences similarity measures in a column chart

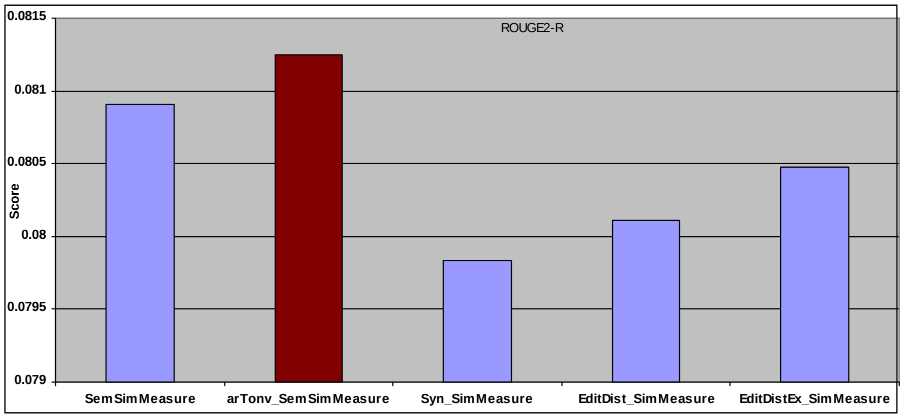


Figure 4.18: ROUGE2 scores showing the performance for the different sentences similarity measures

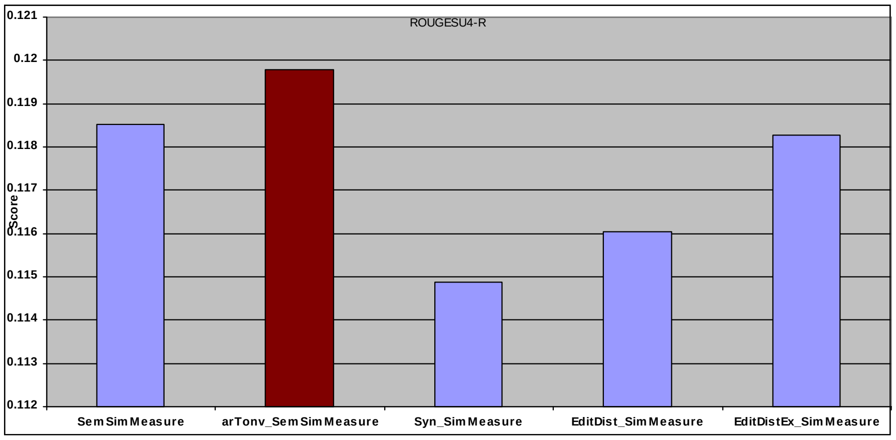


Figure 4.19: ROUGESU4 scores showing the performance for the different sentences similarity measures

The obtained results in the shown column charts support the idea that introducing different semantic-based measures can lead to performance improvement. For instance, the measure *Syn_SimMeasure* successfully captures the similarity between the words "*resolution*" and "*settlement*" in the two sentences: "*The defendant reached a settlement with the plaintiff by paying 20 million dollars*" and "*The defendant reached a resolution with the plaintiff by paying 20 million dollars*". On the other hand, replacing one of the two words with "*agreement*" would cause a failure in capturing the similarity between the words. Instead of performing simply words matching, a more refined measure is used with *SemSimMeasure* and *arTonv_SemSimMeasure* as they both utilize *JCn*'s metric when computing the similarity between words. The obtained results of these two measures in the column charts highlight this observation.

It was noted in the *arTonv_SemSimMeasure* that some adverbs are transformed to verbs, while their corresponding words in other sentences are of type noun. Since the similarity computation between words is performed only on the same POS due to the WordNet limitations mentioned in the previous sections, the benefits gained from the adjectives/adverbs transformation process is therefore dependent on the POS of the words processed in different sentences. When computing the semantic distance between two sentences, it would be best to devise a way for comparing every term from the first sentence with every term in the second regardless of their POS. In part, this is reflected in the performance of the *EditDistEx_SimMeasure* in ROUGE1 which obtained better results than *SemSimMeasure* even though it implements simple words matching while the later utilizes JCn's metric but ignores adjectives/adverbs. In the next chapter, I propose a Wikipedia-based semantic relatedness measure that takes into account every term from the two compared sentences regardless of their POS.

When examining the summaries generated by the best performing summarizer, it can be noted that the grammar of the sentences are acceptable. This is expected as the summarizer is extractive. However, there seems to be an issue with the redundancy of some sentences within some of the generated summaries. In an attempt to address this issue, I opted to include a redundancy and diversity checking layer in the post-processing stage of the summarizer. The next section provides more details about the theory behind this, the implementation and evaluation results.

4.4 Enhancing Diversity and Reducing Redundancy

In this section, I argue that an effective summarization system should take into consideration the following factors:

- **Diversity:** The summary should contain sentences which are as much diverse or different from each other as possible. This should ensure that the summary would contain the largest amount of diverse and important information.
- **Redundancy:** The summary should contain as few redundant sentences as possible. This may have a similar effect to Diversity in some cases, but is computed differently in my system as described in the next sections.

- **Coverage:** The summary should cover all the important and relevant points in the original source document. In the built system, user queries are used to drive the focus of the generated summaries and hence, the coverage is affected by what the user chooses.

Different versions of the built summarizer were implemented with different changes made mainly in the post-processing stages. The system variations were compared with each other to test the effects of introducing new features to the system. The following sections explain the different variations of the system and the main differences between them. To show the effect of introducing each of the three factors: Redundancy, Diversity and Coverage, I used the best run during the previous evaluation as the baseline, namely *arTonv_SemSimMeasure*. The other variations of the summarizer add a redundancy/diversity checking layer in the post processing stage. The layer utilizes the previously defined sentences similarity measures that were described in section 4.2.2.2 for this purpose. An evaluation performed at the end reports the difference performance caused by the introduction of each measure.

4.4.1 The Baseline

The core of the baseline summarizer uses the same components developed in section 4.2. Each sentence is given a score representing its importance based on the extracted features as represented in Equation 4.2. No redundancy or diversity checking takes place with this approach.

When computing the semantic similarity between two sentences (or a sentence and the query/title), words of the same POS are compared with each other. In particular, the system compares nouns from the first sentence with nouns from the second, and verbs from the first with verbs from the second. If an adverb or an adjective is encountered, they are converted to their corresponding nouns/verbs if possible with the help of WordNet relations.

4.4.2 Redundancy-Syn

First, the scores of all sentences are computed as in the Baseline. Then, the document terms are expanded with their synonyms using WordNet. After completing the processing stage and forming a list of ranked candidate sentences to be added to the summary for the first time, I add only the highest ranking sentence to the summary. Then, before adding any of the next highest scoring sentences to the summary, I check the similarity between the candidate sentence and the summary sentences using the *Syn_SimMeasure* metric described in section 4.2.2.2. Only if the similarity scores are below a previously specified threshold it is added to the summary. In effect, this additional process performs redundancy checking to ensure that sentences added to the summary do not contain redundant information to what is already in the summary.

4.4.3 Redundancy-Sim

After scoring all sentences in the processing stage using the baseline approach, I apply a redundancy checking process using the previously described metric: *arTonv_SemSimMeasure*. With this approach, the semantic similarity score is computed between every two sentences nouns and verbs (nouns vs. nouns and verbs vs. verbs). For computing the similarities between words, I adopt the semantic similarity measure Sim_{jcn} [47] which was illustrated in Expression 3.4.

The redundancy checking is applied here in a post-processing stage in a similar fashion to what is performed in the *Redundancy-Syn* approach. Only the top scoring sentence is added to the summary first. Subsequently, Sentences are added to the summary only if the similarity score between the candidate sentence and the summary sentences are below a pre-determined threshold.

4.4.4 Diversity-Ant

As with the previous methods, all sentences are scored first using the baseline method. The metric of the class *Ant_SimMeasure* described in section 4.2.2.2 is used for computing the diversity of every candidate sentence against the sentences already in the summary. After adding the top scoring sentence to the summary, only the candidate sentences with diversity scores larger than a previously determined threshold are added to the summary. The diversity scores are computed by applying expression 4.3 for each candidate sentence against all the sentences that were already added to the summary.

4.4.5 Diversity-Sim

This approach is similar to Diversity-Ant but differs in using *Ant_SemSimMeasure* and Expression 4.4 instead of *Ant_SimMeasure* and Expression 4.3.

4.4.6 Levenshtein Distance

The edit distance method that was described for the models *EditDist_SimMeasure* and *EditDistEx_SimMeasure* are implemented to check the redundancy of sentences in the post-processing stage.

4.4.7 Experiment

I implemented a summarizer with the above-mentioned variations. The setup of the system is similar to the setup for my summarization system used to participate in TAC08 [43] and described in section 4.2. After computing the semantic similarity score between any two words, I compare that score with the predefined threshold. I set the semantic similarity threshold to 0.7 which I found to be optimal in my previous experiments [153] for the used dataset. To optimize the system's running time, I had used a previously built matrix containing the semantic similarity scores for all nouns extracted from WordNet. In all of the system variations α was given the value of 0.75 to give more weight to the topic statements of each data set than to the headlines. The topic of each dataset was used as the user query.

4.4.8 Test Data and Evaluation Results

To evaluate the system, I ran the different system variations on the TAC08 datasets and computed the ROUGE scores for each variation. For each run, three scores were computed: ROUGE1, ROUGE2 and ROUGESU4. The ROUGE measure is widely used for evaluating summaries in the NLP community. With it, the summary quality is measured by counting the number of overlapping units (or word sequences) between a predicted summary and the generated summary by the system.

A summary of the retrieved results is shown in Table 4.5 and comparisons between the scores are shown in Figure 4.20, Figure 4.21, and Figure 4.22. Each row represents the results of one run starting with the Baseline. *Red-Syn* refers to the run where the redundancy between all sentences was computed after expanding their terms with their synonyms. The rank of each sentence is affected by its redundancy score. For a high ranking sentence A, right before it is added to the summary, if another sentence B already exists in the summary and has a high redundancy score with sentence A, it is decided not to add sentence A to the summary. This logic applies for the rest of the implemented redundancy metrics. *Red-Sim* refers to Redundancy with Semantic Similarity computation. As for *Red-LevDist*, it refers to Redundancy which takes into account the previously described Levenshtein Distance, where *Red-LevDist-Exp* refers to its expanded version. As for *Div* and *Ant*, they refer to Diversity and Antonyms respectively.

Evaluation	**ROUGE1**	**ROUGE2**	**ROUGESU4**
Baseline	0.35801	0.08125	0.11979
Red-Syn	0.35840	0.08136	0.11991
Red-Sim	0.35956	0.08218	0.12038
Red-LevDist	0.35870	0.08176	0.11987
Red-LevDist-Exp	0.36022	0.08338	0.12075
Div-Ant	0.35788	0.08128	0.11982
Div-Sim	0.35876	0.08218	0.11990

Table 4.5: **ROUGE Evaluation Results of the Different Variations of the WordNet-based System**

As shown in Table 4.5 and Figure 4.20, the obtained results suggest that introducing the different metrics improved the overall performance of the system. The only exception is for the run *Div-Ant* which gave an inferior performance when compared against the Baseline in ROUGE1. By examining Figure 4.20, Figure 4.21 and Figure 4.22, one can see that the run *Red-LevDist-Exp* gave the best performance. The next best run is *Red-Sim* followed by *Div-Sim* which gave a comparable performance to the former in ROUGE2.

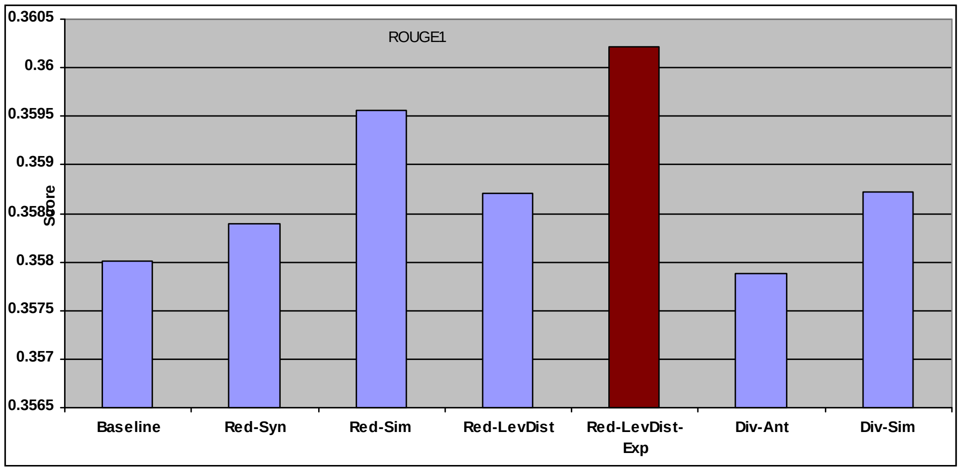


Figure 4.20: **ROUGE1 scores showing the effects of redundancy and diversity checking module in WordNet-based summarizer**

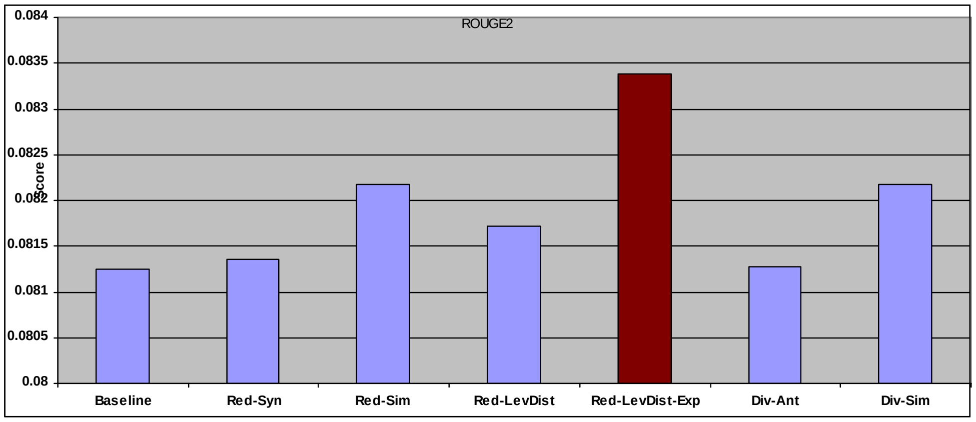


Figure 4.21: ROUGE2 scores showing the effects of redundancy and diversity checking module in WordNet-based summarizer

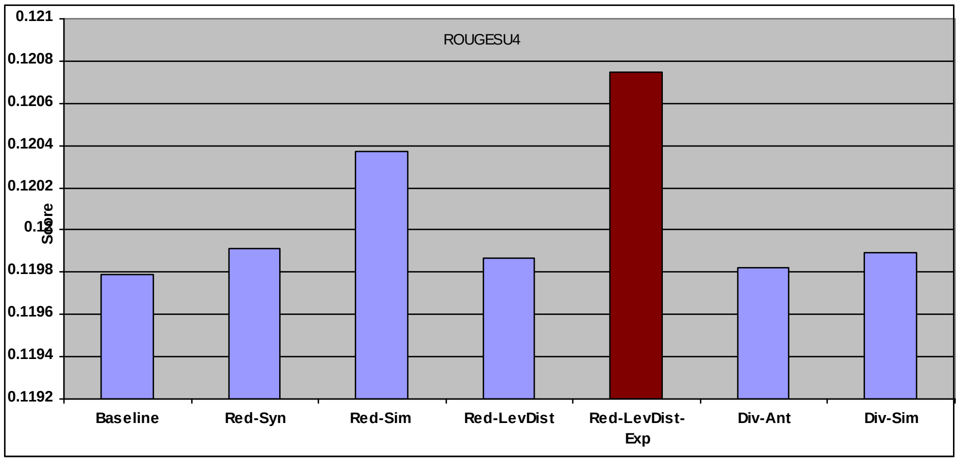


Figure 4.22: ROUGESU4 scores showing the effects of redundancy and diversity checking module in WordNet-based summarizer

It can be noted from the obtained results that measures that employ JCn's semantic similarity exceed in performance their counterparts which simply check for words overlaps and apply synonyms expansion. Also can be noted is the effect of introducing Antonyms to the system in the runs labelled *Div-Ant* and *Div-Sim*. In the former, diversity checking was enforced by strictly comparing whether the antonyms of the words in a sentence match the words of another sentence. This comparison does not seem to be always effective especially in cases when two words carry some degree of contrast meaning but the antonyms of one do not yield a match for the other. On the other hand, *Div-Sim* seems to be more flexible and captures strict antonyms in addition to those showing some degree of contrast. An example for this is the words *concealed* and *expressed* in the two sentences "The father *concealed* his illness from his family" and "The father *expressed* his troubles to the doctor". The two words are not antonyms in WordNet but appear to have a high degree of contrast. Enforcing a diversity checking measure allows for inclusion of new information and different sentiments in the summary. In some cases, encountering auto antonyms may negatively affect how the diversity detection module works. Take the word "overlook" as an example. Depending on the context it is placed in, it can sometimes mean "to inspect" while in other instances mean "fail to inspect". Taking the antonym of the word without factoring in the context it is placed in may lead to wrong conclusions. This issue is implicitly addressed in the Wikipedia-based summarizer I describe in the following chapters. With Wikipedia, the methods I rely on factor in indirectly the context of a term while computing the similarity between sentences or text fragments as will be described in the following chapters.

To get a better idea on how the variations of the system, I examined sample summaries generated by the different implemented methods. I start by a summary generated by the baseline summarizer for document set D0801A taken from the TAC08 documents collection. In Figure 4.23 I show the system summary in addition to several reference summaries written by humans for the same document set.

Human Reference Summaries
(1) The European Airbus A380 flew its maiden test flight from France 10 years after design development started. The A380 super-jumbo passenger jet surpasses the Boeing 747 and breaks their monopoly. Airlines worldwide have placed orders but airports may need modification to accommodate the weight and width of the A380. U.S. airlines have not placed an order. Airbus has fallen behind in production and a backlog of orders has developed. Airbus must sell at least 250 planes to break even financially. The A380 is overweight and modifications to meet the weight requirements impacted the budget. Additional test flights are planned.
(2) Emirate Airlines ordered the first passenger A380 five months before its December 2000 launch. In January Federal Express ordered the first cargo A380. Thirteen non-American airlines have placed 154 orders; China and Hong Kong have options. Commercial deliveries begin first quarter 2006 to Singapore. A380s will land at 25 airports worldwide, including New York, Los Angeles, San Francisco, Miami, Chicago, Dulles, Memphis and Anchorage. In February 2001 Airbus's Hamburg plant expanded. Toulouse production started in January 2002. In July 2003 Broughton, Wales got an Airbus plant. The first A380 arrived in January 2005, taking its maiden flight April 27.
(3) The largest passenger airliner ever built, the Airbus 380(A380), took off on its maiden four-hour flight on April 27, 2005 in France. The European company, Airbus, is the newest competitor with the Boeing Company. The A380 is designed to carry 555 passengers, but can be expanded to 800 seats. Airbus stresses the plane's fuel efficiency. Its first test flight was successful. Orders for 149 aircraft from airlines and freight companies have been received. No US airline has ordered the jet yet. First commercial deliveries to Singapore Airlines are scheduled for 2006.
(4) In 1994 Airbus began engineering the A380, a superjumbo airliner larger than Boeing's 747. Component production started in 2002 in Germany and France. A Toulouse, France assembly line opened in 2004. Parts were pared down and new materials introduced to keep the plane at target weight but sent the plane over budget. The A380, carrying between 555 and 840 passengers, was unveiled in January 2005 and test flown in April. Airports need to make design changes to accommodate this overlarge plane that boards on two levels. The US objects to government subsidies to Airbus and airport neighbors complain about noise.
System Summary
(1) The 787, which was launched a year ago, is scheduled to enter service in 2008. **(2)** European airplane maker Airbus "is likely to discuss before the end of the year" a possible increase in production capacity of its new super-jumbo A380 aircraft, Airbus' production chief Gustav Humbert said in a magazine interview released Tuesday. **(3)** The superjumbo Airbus A380, the world's largest commercial airliner, took off Wednesday into cloudy skies over southwestern France for its second test flight. **(4)** The Airbus flagship is due to enter service next year. **(5)** If major airports are slow to support the new plane, airlines may hesitate to buy.

Figure 4.23: Summary generated by the baseline for document set D0801A in the TAC08 documents collection

As can be noted from the above summary, there appears to be some redundancy between sentences 1 and 4 in the system summary. In an attempt to handle this redundancy, several measures have been applied. With the redundancy and diversity checking measures applied, the system checks the redundancy and diversity of the top candidate sentences before adding them to the summary. With the first redundancy checking measure implemented Red-Syn, the effect of this for document set D0801A is shown in Figure 4.24.

(1) The 787, which was launched a year ago, is scheduled to enter service in 2008. **(2)** European airplane maker Airbus "is likely to discuss before the end of the year" a possible increase in production capacity of its new super-jumbo A380 aircraft, Airbus' production chief Gustav Humbert said in a magazine interview released Tuesday. **(3)** The superjumbo Airbus A380, the world's largest commercial airliner, took off Wednesday into cloudy skies over southwestern France for its second test flight. **(4)** Airbus is hoping the 550-seat A380, the world's biggest jet, will revolutionize air travel.

Figure 4.24: Summary generated with the redundancy checking measure that expands words with synonyms and applies simple words matching for document set D0801A

This measure is simple in the sense that it does not compute the semantic similarity between compared words but only checks the lexical form of the different terms after expanding them with their synonyms.

Another redundancy checking measure I applied is Red-Sim which computes the semantic similarity between words when performing the checking. An example for a summary generated after applying this measure is shown in Figure 4.25 where summary A represents the generated summary before redundancy handling while summary B is generated after checking for redundancy.

Reference Summaries

(1) Ice continues to melt at an alarming rate in both the Arctic and Antarctic. Higher temperatures have shrunk the Arctic ice area 10% and its thickness 42% in 30 years. The permafrost is shrinking, endangering infrastructure. These changes are threatening the culture and economy of the indigenous Artic population. Ice shelves in the Antarctic are collapsing. The melting of the West Antarctic Ice Sheet could raise ocean levels worldwide approximately 15 feet. Increased tourism in the Antarctic is having an environmental impact. Researchers are debating whether greenhouse gases or natural climate cycles are the biggest cause of the melting.

(2) Collapse of coastal Antarctic ice shelves accelerated eight-fold the seaward flow of inland glaciers, raising sea levels: Larsen A (1995), Wilkins (1998), Larsen B (2002), Larsen C (this century). Currents undermine the Ross and Ronne ice shelves, enabling ice flows from deep within the West Antarctic ice sheet. Arctic permafrost thawed; glaciers and sea ice retreated. In 30 years the Arctic ice cap's area shrank by 10%, its thickness by 42%, opening shorter maritime routes when Arctic sea ice disappears in future summers. Siberian lakes disappeared. Indigenous cultures and glacier tourism suffered. Bird migrations shifted. Northern Hemisphere weather will worsen.

(3) In Antarctica and the Artic, ice melts are causing complex questions about the impact of global warming. In Antarctica huge glaciers are thinning and ice shelves are either disintegrating or retreating. These findings are possible indications of global warming. Information gathered about Antarctica coincides with a recent report on accelerating climate changes in the Arctic. A Chinese scientist predicted that the Artic icecap would melt by 2080. The Arctic's indigenous people (about 4 million) are fighting global warming because it will be a threat to their societies, economies and culture.

(4) The thinning of glaciers and ice shelves, as well as the softening of the permafrost, has accelerated greatly in recent years. While it is not certain that the man-made greenhouse effect is entirely to blame, it is clear that man must take steps now to address the problem. Global warming affects everything: oil-platforms, the society of peoples who are indiginous to the polar regions, polar animals, migratory birds, lakes (which are drying up as the permafrost melts), and even tourism. As melting cold fresh water enters the salty sea, it will affect ocean currents and therefore world climate.

System Summaries

A

(1) The collapse of a huge ice shelf in Antarctica in 2002 has no precedent in the past 11,000 years, according to a study to be published on Thursday that points the finger at global warming. **(2)** Zhang Zhanhai, director of Polar Research Institute of China, said that the melting rate of Arctic ice is alarming. **(3)** Zhang said, the cold front that affects China mainly comes from Siberia, but the source of the cold front is Arctic. **(4)** The melting of Arctic ice will not only be a sign of threat, it is also a good news, Zhang said.

B

(1) The collapse of a huge ice shelf in Antarctica in 2002 has no precedent in the past 11,000 years, according to a study to be published on Thursday that points the finger at global warming. **(2)** Zhang Zhanhai, director of Polar Research Institute of China, said that the melting rate of Arctic ice is alarming. **(3)** Zhang said, the cold front that affects China mainly comes from Siberia, but the source of the cold front is Arctic. **(4)** They reported in the journal Science last September that a half-dozen glaciers there are now thinning and accelerating.

Figure 4.25: Summaries generated for document set D0802A before (A) and after (B) applying the redundancy checking measure that computes the semantic similarity between words

When examining sentences 2 and 4 of summary A, it can be noted that terms such as *threat* and *alarming* are not synonyms and would not be handled by the measure applying only synonyms expansion. When considering the semantic similarity between the terms in the redundancy checking stage, I obtain the summary *B* which has sentence 4 from summary *A* replaced by another sentence..

The edit distance measures I applied are especially useful when encountering terms that do not exist in WordNet or recognized as a NE. It is also useful for detecting acronyms and new words in specific domains. When encountering terms such as house3 and house1, it is able to correlate between the two. An example reflecting the effect of introducing this measure is presented in Figure 4.26 where summary *A* is generated before applying the redundancy checking while summary *B* is generated after.

Reference Summaries
1 After 18 months the International Astronomical Union has still not determined the criteria for planethood. Pluto's eventual status will also define new space objects. However the data from the Pluto space probe in 2015 will come too late to settle this argument. Recent discovery of two additional moons around Pluto complicate the issue. The diameter of Pluto and UB313 is about 1,380 and 1,800 miles respectively and both have methane ice on their surfaces. Proposed definitions based on size or gravitational pull either include or exclude Pluto and UB313. The scientific deadlock at the Union will not be resolved quickly. 2 A committee of the International Astronomical Union (IAU) is considering the question of planethood of Pluto and 2003 UB313 (Xena). Both might be called "minor planets" or could be considered KBOs. NASA's Hubble Space telescope strengthened Pluto's claim to planethood by finding its second and perhaps third moon. On Jan 20, 2006 NASA launched its New Horizon spacecraft to explore Pluto and the Kuiper Belt which it is scheduled to reach in July 2015. Meanwhile German astronomers determined that the diameter of 2003 UB313 is 1800 miles compared to Pluto's 1380. The IAU committee continues to deliberate. 3 Debate continues over what constitutes a planet. Some astronomers say a planet should have a diameter greater than 2,000 kilometers (both Pluto and UB313 do). Another suggestion is roundness: this would include these two and likely several others yet unknown. Another possible definition is that a planet must gravitationally dominate its surroundings, which would exclude Pluto and UB313. Pluto became more interesting when the Hubble telescope discoverd that Pluto had not just one but three moons. Scientific interest in Pluto and Kuiper Belt objects led to the New Horizons spacecraft mission: a nine year trip to Pluto and beyond.

4
Scientists debate redefining "planet" or retiring the term. KBO 2003 UB313, discovered by Michael E. Brown, was confirmed by surface heat emanations to be 30 percent larger than Pluto. Both have surface methane ice. If Pluto is a planet, 2003 UB313 should be too. The Hubble Space Telescope showed two more small moons orbiting Pluto, 30-100 miles in diameter. Pluto and its large moon Charon, close physically and in size, may be a binary planet system orbiting a center mass between them. At least four KBOs have known moons. Pluto would be the first KBO to have multiple satellites.

System Summaries

A
(1) Pluto, the smallest and most distant planet of the solar system, may have three moons instead of just one, according to images NASA said were taken by the orbiting Hubble Space Telescope. **(2)** The discovery of 2003 ub313 has intensified a debate over the definition of a planet, which has swirled about Pluto since the late 1990s. **(3)** The jury is still out on the impact additional moons will have on the ongoing debate over whether Pluto is actually a planet. **(4)** The debate intensified this summer with discovery of a Kuiper Belt object, UB313, that is larger than Pluto.

B
(1) Pluto, the smallest and most distant planet of the solar system, may have three moons instead of just one, according to images NASA said were taken by the orbiting Hubble Space Telescope. **(2)** The discovery of 2003 ub313 has intensified a debate over the definition of a planet, which has swirled about Pluto since the late 1990s. **(3)** The jury is still out on the impact additional moons will have on the ongoing debate over whether Pluto is actually a planet. **(4)** Possessing a moon is a not a criteria of planethood since Mercury and Venus are moonless planets.

Figure 4.26: Summaries generated for document set D0811B before and after applying the edit-distance-based redundancy checking measure

It can be noted that the original summary A contains the term *UB313* in sentences 2 and 4. The term does not exist in WordNet and is not recognized as a NE and is thus handled by Red-Lev-Dist measure. This results in generating summary B in which sentence 4 is replaced with another sentence. Note that the term *planethood* in the new sentence does not exist in WordNet too and would be handled by Red-Lev-Dist if a redundancy is detected. Another example showing the effectiveness of the edit distance measure is when encountering misspellings or typos in text. An example illustrating this is shown in Figure 4.27.

Reference Summaries

1
The directors of Fannie met on Dec. 19 to decide Raines' fate, but the meeting concluded without any announcement of a decision. On Dec. 21 Raines resigned with his departure structured as an early retirement. The directors' decision in favor of Raines' retirement rather than dismissal would reap him $8.7 million in deferred payments and an annual pension of more than $1 million. On Jan. 21, 2005, however, Fannie announced that Raines would be denied the cash bonus that he would have received for 2004.

2
By October 12, 2004 the U.S. Attorney's Office in Washington was investigating Fannie Mae CEO Franklin Raines, who denied wrongdoing, after a caustic September 22 Office of Federal Housing Enterprise Oversight report vetted by the SEC, charged that Fannie Mae violated Financial Accounting Standard 133 and FAS-91, inflating Fannie Mae's reported net earnings by $9 billion, so that Raines and other top executives received maximum bonuses of $27.1 million vice no bonuses, and got higher, allegedly manipulated, prices for their own sales of Fannie Mae stock. The Justice Department told Fannie Mae to preserve documents related to OFHEO's report.

3
Raines testified to Congress that he expected to be held accountable if his interpretation of the accounting rules was not accepted. The chief accountant of the SEC ruled that Fannie Mae violated accounting standards overstating its profits by 38 percent since January 2001. Raines is seeking a vote of confidence from the directors as he struggles to survive criminal investigation. Raines retired under pressure from federal regulators. Raines will be denied bonuses for 2004 as Fannie Mae eliminated bonuses for top executives.

4
The SEC ruled that Fannie Mae had overstated its profits by 38 per cent in the years 2001-2004. The OFHEO held Raines responsible the company's emphasisis on earnings over accuracy and demanded significant changes in senior management. On 21 December, Raines took early retirement with millions of dollars in benefits and stock, plus an annual pension of over one million dollars. The company still faced a Justice Department criminal investigation, an SEC civil investigation, further OFHEO examination of its accounting, and several class-action lawsuits. As of 21 January, Fannie Mae was withholding millions in bonuses for top executives.

System Summaries

A
(1) Mortgage giant Fannie Mae is now the target of a formal inquiry by the Securities and Exchange Commission over its accounting practices. **(2)** The company remains under investigaton by the SEC, the Justice Department and OFHEO, which is examining additional accounting issues. **(3)** Chairman and chief executive Franklin Raines and chief financial officer Timothy Howard said Fannie Mae did nothing wrong in its accounting and insisted that the regulators' allegations represent an arguable interpretation of complex rules. **(4)** In its filing with the SEC, the company said it would cooperate fully with the probe.

B
(1) Mortgage giant Fannie Mae is now the target of a formal inquiry by the Securities and Exchange Commission over its accounting practices. **(2)** The company remains under investigaton by the SEC, the Justice Department and OFHEO, which is examining additional accounting issues. **(3)** Chairman and chief executive Franklin Raines and chief financial officer Timothy Howard said Fannie Mae did nothing wrong in its accounting and insisted that the regulators' allegations represent an arguable interpretation of complex rules. **(4)** For Raines and Howard, the stakes go beyond keeping their jobs.

Figure 4.27: Summaries generated for document set D0810B before and after applying the expanded edit-distance redundancy checking measure

In the given example above, I have the word *OFHO* which is not handled by WordNet or recognized as a NE. I also have the misspelled word *investigaton* appearing in sentence 2 of summary A. In sentence 4, the word *probe* is one of the synonyms of investigation.

After enforcing the expanded edit-distance redundancy checking measure, I have the summary B generated.

Diversity checking was another aspect that was implemented in the system. For checking the diversity between sentences, antonyms contained within each sentence are considered. An example for this is shown in Figure 4.28.

Reference Summaries

1
Congressional hearings and an undercover probe investigating illegal steroid use in Major League Baseball and other sports continues. Baseball Commissioner Bud Selig admitted the League's policy is not strong enough and has proposed much stricter penalties. They include a 50-game suspension for the first offense and a lifetime ban for the third one. These penalties would apply to minor-league players next year. Selig also requested amphetamines be added to the banned substances. Congress has threatened to implement a testing system unless the League strengthened its policy. The Players Association has not responded to the proposed penalties.

2
Under pressure from Congress, Major League Baseball's players association toughened drug testing rules and penalties again for the 2006 season. Suspensions will be increased to 50 games for a first offense, 100 for a second, and a lifetime ban for the third. Congress launched an undercover probe into illegal steroid use in Major League Baseball. News of the probe surprised MLB officials. Rafael Palmeiro was suspended 10 days for a positive steroid test. He denied intentionally taking the drug. Other players with 10-day steroid test suspensions were Felix Heredia and Carlos Almanzar. New doping allegations surfaced about Barry Bonds.

3
Congress has launched an undercover probe into illegal steroid use in major league baseball. To highlight the problem, baseball heavy hitters Mark McGuire, Raphael Palmeiro and Sammy Sosa testified before the responsible congressional committee. In a letter to the players' union, Selig proposes a 50-game suspension for first offenders instead of the current 10-game suspension. A second offense would result in a 100-day suspension and a third, a lifetime ban. The players' union has concerns since several steroid agents stay in the blood stream for a long period. A one-time user who swore off could show positive in future testing.

4
Congressional investigators chided Major League Baseball leaders for being uncooperative regarding steroid use and urged star players to take public responsibility for their actions. Congress also launched an undercover probe. Players proposed strengthening the collective bargaining steroid proposal. Bud Selig proposed a 50-game suspension for first offenders, 100-game suspension for second, and lifetime ban for third. 2003 tests showed 104 steroid users. Eleven players were suspended for steroids in 2005. Felix Heredia, Carlos Almanzar and Rafael Palmeiro tested positive. BALCO's Victor Conte and James Valente, Bonds trainer Greg Anderson, and chemist Patrick Arnold were charged or sentenced concerning steroid distribution.

System Summaries

A
(1) The Government Reform Committee was hearing from six subpoenaed players, including Mark McGwire and Sammy Sosa, along with commissioner Bud Selig and other baseball executives, medical experts and the parents of two amateur athletes who committed suicide after taking steroids. **(2)** Baseball's revised steroid testing program has suspended four players in two months. **(3)** In a letter to the players' union, baseball commissioner Bud Selig has admitted that the newly revised steroid policy is not enough. **(4)** Last week, Selig wouldn't commit to investigating. **(5)** The 40-year-old Palmeiro is hitting. **(6)** Steroids users cheat the game.

B
(1) The U.S. government has stepped in to investigate steroid use among professional baseball players use because it felt the sport wasn't doing enough to enforce its own policies. **(2)** Baseball's revised steroid testing program has suspended four players in two months. **(3)** In a letter to the players' union, baseball commissioner Bud Selig has admitted that the newly revised steroid policy is not enough. **(4)** The Government Reform Committee was hearing from six subpoenaed players, including Mark McGwire and Sammy Sosa, along with commissioner Bud Selig and other baseball executives, medical experts and the parents of two amateur athletes who committed suicide after taking steroids.

Figure 4.28: Summaries generated for document set D0835B before and after applying the diversity checking measure

In the given example above, it can be noted that summary A contains the term *Amateur* in sentence 1. The term *Amateur* has the antonyms *professional* and *pro*. After applying the diversity checking measure, the last three sentences of summary A are replaced with a sentence containing the term *professional*. Applying this method is not always error-free. When examining some other documents such as D0821DA, it can be noted that some terms which exist in that document carry more than one meaning such as the term *feet*. It can correspond to the unit of length or the part of the leg of human beings (plural of foot). The context the term appeared in is “70 feet down”. The summarizer interpreted this incorrectly and considered the word *head* as its antonym during the diversity checking stage.

To get even a better view of the performance of my summarizer, I examined the performance of other summarizers built by other TAC08 participants. I found that three other participants had algorithms relying on WordNet for summarization. The performance results of these systems are illustrated in Table 4.6. I used ROUGE2 and

ROUGESU4 since they have the strongest correlation with content responsiveness score which is assigned by human judges and measures the information coverage of the summaries. In addition, the averages of these scores were provided to all participants for reference and they are practically ready to be compared against my best performing system.

Table 4.6 shows the scores for my system, the best system and worst system of the 71 participants in TAC08. In addition, the average scores of all participants are shown in the table. The systems abbreviated with *NG Graph*, *SMUST* and *NESS* all rely on WordNet for summarization. The system labelled *NG Graph* used WordNet for query expansion looking up all query words in WordNet and appending the “overview of senses” results to the query [154]. The algorithm of the system represents texts by using n-grams positioned within a context-indicate graph, hence the name n-gram graphs. Important sentences are judged by comparing the graph representation of each sentence with the graph representation of the expanded query. The system authors claim that the poor performance obtained by their system is due to “noise” or non important/relevant terms appended to the query during the query expansion process.

The system labelled SMUST [155] used the TextRank [79] algorithm to extract topic terms from the source documents and assign a score to each topic. Features such as detected topics, sentence position and overlap with topic terms were detected and assigned scores in each sentence. The Path similarity measure was used to determine the similarity between terms in different sentences and the topic terms. A linear combination of the scores was used to generate a score for each sentence.

Evaluation	**Scores**		**Ranks**	
	ROUGE2	**ROUGESU4**	**ROUGE2**	**ROUGESU4**
NG Graphs	0.03484	0.07657	71	71
SMUST	0.06203	0.09974	56	57
NESS	0.08200	0.11881	23	22
AUEB	0.09621	0.13434	4	4
My Summarizer	0.08338	0.12075	20	19
Average	0.07293	0.11075		
Worst System	0.03343	0.06517		
Best System	0.10382	0.13646		
Variance	0.00022	0.00023		
Standard Deviation	0.01477	0.01520		

Table 4.6: ROUGE scores of my System and other participants in TAC 2008

The summary is finally formed by choosing the highest scoring sentences. The system labelled *NESS* [156] selects sentences based on linguistic metrics such as TFIDF scores that measure the relevance of sentences to the source documents topics. It uses WordNet to conduct topic-expansion by extracting the synonyms of each noun in the topic from WordNet and append them to the topic. Just like the other systems, each sentence is given a score based on its weighted features and the summary is generated by choosing the highest ranking sentences.

The implementation of my system is similar to the mentioned above in that each sentence is assigned a score based on the combination of its weighted features. However, it differs in some of the chosen features such as the inclusion of Named Entities and similarity of a sentence against the rest of sentences in a document. In addition, I rely on JCn's metric for computing the semantic similarity between words and devised my own measures for computing the similarity between sentences. Redundancy and diversity checking are additional stages implemented in my system. The most comparable system to mine performance-wise is *NESS*. However, it applies a complex level of syntactical processing through their FIPS [157] parser. My system does not rely on the analysis of syntactical structures of sentences and is much simpler than theirs. As for the other two systems, my system performance appears to be more competitive than theirs as shown in the table.

The *AUEB*'s summarizer [158] uses a Support Vector Regression (SVR) model [159] to rank the summary's candidate sentences. For training the SVR, DUC 2006 documents were used to construct training vectors for each sentence. The features used are sentences position, number of Named Entities, Levenshtein distance between sentence and query, word overlap with query and content words frequencies. Even though this summarizer relied on basic features, the results it achieved are among the best for the update summarization task. The summarizer I built uses all of the features this system relies on, and does not require any training. Due to the limited scope of WordNet, it is possible that some of the important content terms that define relations between different sentences do not exist in WordNet and thus are ignored in my system. As for AUEB's summarizer, it is expected that its performance will suffer if used with other types of datasets without any further training.

4.5 Conclusion

In this chapter, I described the structure of my WordNet-based summarizer and its main components. I defined several measures for computing the similarity between sentences and highlighted how WordNet was used to expand these measures. I participated in the TAC2008 update summarization task with two runs utilizing one of the implemented similarity measures and reported the obtained results. In an experiment, I used these defined measures in the built summarizer to test their effects on the performance of the system. After observing the performance improvements, I attempted to enhance the performance even further by introducing a module for reducing redundancy and enhancing diversity within the generated summaries. The results of the evaluations performed reflect varying levels of improvements to the system caused by the addition of the module.

While it does seem from the results above that the introduction of WordNet-based semantic similarity features gave a positive effect to the performance of the system, some work on the base system and subsequently the rest still needs to be revised for a better improvement. Even though WordNet gave competitive results in the performed experiments and in others' too [160], using it as the main backend source to summarize documents may not always be the most optimal solution. WordNet was manually built and covers a limited number of concepts that may not include some aspects in many domains. It is therefore useful to consider using other thesaurus such as Wikipedia which is constantly being updated and is the world largest encyclopaedia. Some slang words and domain specific terms may not be exist in WordNet and documents containing such terms may not be suitable to be used with the developed system. Another aspect that should be noted is the limitation of applying the above described methods and algorithms to only English documents. This is mainly due to having Wordnet as monolingual. While other projects such as EuroWordNet have attempted to expand the WordNet database to include several other languages, the work completed is still very limited and span a relatively small number of languages. Human experts, time and additional resources would be required to extend such work and produce high-quality databases for tens of other languages. And then even if created, updating the databases with new entries would still be costly and likely to be time lagging between the current state of the language and the how it is represented in the database.

In an attempt to address the above limitations, I used another repository, namely Wikipedia, which is orders of magnitude larger than WordNet. For the sake of comparison, note the number of articles it has which is over 3 million, which each can be treated as a concept, against 117,659 synsets contained in WordNet. Wikipedia is run by a large number of volunteers, reducing the running cost in comparison with other repositories such as WordNet. Updating Wikipedia to include new events and concepts is usually performed in a short period of time after the event occurs. However, one of the main advantages of WordNet over Wikipedia is in its design making it easy to read and process by machines. In the next chapter, I propose a set of algorithms to allow me to use Wikipedia and exploit its content in a competitively-performing summarization system.

Chapter 5

Summarization Aided with Wikipedia

In this chapter, I describe how I generate features from Wikipedia and then use them in a Wikipedia-based summarization system. Also, I present the evaluation results of my participation in the Text Analysis Conference 2010 (TAC10) Guided Summarization task with the Wikipedia-based Summarization system.

5.1 Overview

An increasing amount of work has been recently applied to enriching text representations for different applications including classification, clustering, information retrieval and clusters labelling. Different kinds of knowledge bases have been used too for the different applications. In [161], WordNet was used to enhance the Classification of text documents by improving the Rocchio algorithm. Rocchio is an algorithm traditionally used in IR and assigns the same importance to training for each class even if it has very few training instances. Their method was supervised and required manual annotations of terms vectors. In [162], WordNet was used for the task of documents clustering. They used WordNet synsets to enrich the representation of documents but without word sense disambiguation. The results they obtained did not show improvements with the use of synsets. In the previous chapter, I reported several methods for using WordNet to aid in summarization by exploiting its relations for improving how semantic similarity is generated.

In this chapter, I describe a methodology for extracting features from Wikipedia and using them in applications such as Automatic Text Summarization and Word Sense Disambiguation. To illustrate the importance of using a large encyclopaedia such as

Wikipedia, I will present a simple example. Suppose that one of the documents to be summarized contains sentences with explicit mention of *Hyperthymesia* (meaning superior memory). For a summarizer employing a BOW method or enriched with low-breadth ontology, a user query such as *Abnormal Psychology* would not be properly processed by the system. The same case applies to humans since an understanding of the meaning of the word *Hyperthymesia* is required to establish a link between it and the *Abnormal Psychology* parts in a document. In WordNet, such a word does not exist and thus WordNet-based systems would not be able to handle it, too. Also, consider the case when two consecutive words such as "*Cat Fish*" provide a new meaning different from the two separate words. With only traditional BOW methods, multi-word concepts are usually misinterpreted or simply omitted. Hence, the use of external knowledge to enrich summarization methods should help address similar scenarios where semantic understanding of the content of the documents and the relationship between its contents and the different queries is needed. In addition, this semantic analysis is especially important when a large number of documents to be summarized are short in length providing less information for training with BOW methods.

Wikipedia was exploited in different Information Extraction and Data Mining applications, but with very limited work being applied in the application of Summarization. In [163], Wikipedia was used to build a thesaurus for use in specific domains. The focus was on using Wikipedia's internal and redirects links for this task with small emphasis on the rich relations and hierarchy available in Wikipedia. In [139] and [164], a method was proposed and evaluated that uses Wikipedia and Open Directory Project (ODP) to obtain representative concepts vectors for documents in the task of text classification. Their idea is similar to mine when building the term-concepts table but without applying the boosting algorithm. In my work, I attempt to leverage the abundant information present in Wikipedia by extracting other features such as strong links and categories structure and integrating them in my system to obtain even better performance.

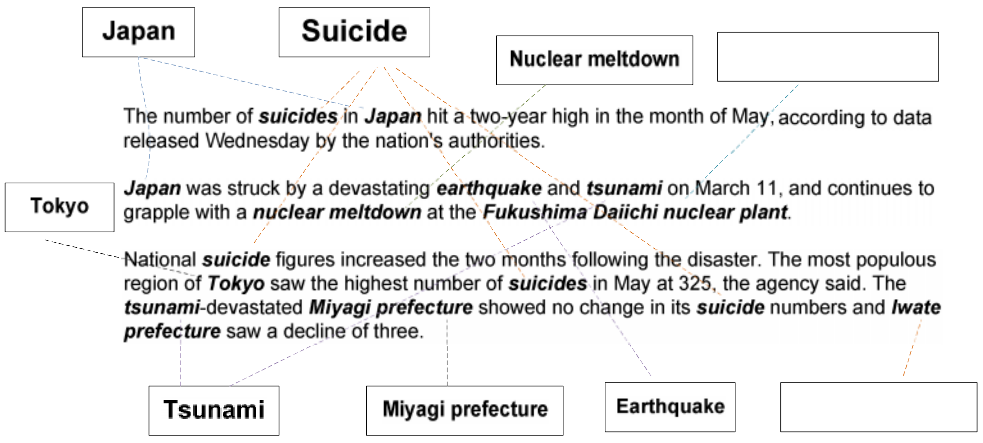
In [165], a methodology was used for detecting the explicitly-mentioned Wikipedia concepts within documents. The authors applied that methodology in the task of documents clustering. With the method I propose, I also consider the related concepts to those explicitly mentioned in the text documents.

In this thesis, I describe a novel framework that utilizes Wikipedia as its underlying knowledge base. The large number of concepts and diverse domains covered in Wikipedia makes it most suitable for the task. Instead of mapping the documents text to a concept or a small group of concepts as done in most of the previous work, I map it to all of the previously-processed Wikipedia concepts. This is achieved by first processing all Wikipedia articles and extracting the relationship between each of its terms and all the concepts existing within Wikipedia. In essence, this forms what I call a *term-concepts table*. Then, I extract the categories structure within Wikipedia and analyze its links. The result of all the mentioned steps (Concepts, Categories and Text) is then combined to form vectors for each group of documents in the preprocessing stage. In the processing stage, the formed vectors are used to decide what the summary candidate sentences are. My experimental results on the TAC10 dataset shows that the system provides competitive results against many others. In addition, the methodologies used are applied to two other applications: Word Sense Disambiguation (WSD) and Documents Classification. I also present the evaluation results for each of these applications.

The rest of the chapter is organized as follows: Section 5.3 describes the main elements of the Wikipedia-based framework. In section 5.4 I show how the extracted features are used in the different implemented heuristics. I follow in section 5.5 by presenting the evaluation results and discuss my findings. Finally, in sections 5.6 I present the usage of the implemented methods in the application of Word Sense Disambiguation.

5.2 Wikipedia-based Framework

My approach relies on the use of the vast and highly organized human knowledge existing within Wikipedia giving it a major advantage over other approaches using smaller thesauruses such as WordNet or Open Directory Project (ODP). Due to its openness and structure, Wikipedia is not suitable for being used directly as is by machines and its content needs to be analyzed first with semantic processing tools. In my semantics-extraction system, I treat each Wikipedia article as a unique Concept and use its title as a label. The content of the article is used to help build a relationship vector between the article terms and its title. The formed term-concepts vector along with the Categories structure existing within Wikipedia and its links are analyzed to aid in computing the relatedness score between any two text fragments. An example for how the extracted features can be used is illustrated in Figure 5.1 where the detected Wikipedia concepts are highlighted. With the strong links method, which will be described in the next subsections, hidden concepts such as “Disaster”, “Nuclear Weapon”, and “Prefectures of Japan” may be detected as well to help identify the most dominant concepts and themes within the document. It is also possible to quantify the relatedness between different documents, sentences or text fragments using the relatedness metrics that will be described in this chapter.



Japan
Suicide
Nuclear meltdown
Tokyo
The number of suicides in Japan hit a two-year high in the month of May, according to data released Wednesday by the nation's authorities.
Japan was struck by a devastating earthquake and tsunami on March 11, and continues to grapple with a nuclear meltdown at the Fukushima Daiichi nuclear plant.
National suicide figures increased the two months following the disaster. The most populous region of Tokyo saw the highest number of suicides in May at 325, the agency said. The tsunami-devastated Miyagi prefecture showed no change in its suicide numbers and Iwate prefecture saw a decline of three.
Tsunami
Miyagi prefecture
Earthquake

Figure 5.1: A document marked with associated Wikipedia concepts

There are a number of stages the framework system has to go through before generating the Wikipedia-based vectors. First, I preprocess the available Wikipedia data to retain its articles text, titles, links and categories structure and remove non-relevant information such as the edit history, image descriptions, and articles authors. Afterwards, I apply some filtering metrics to extract the important concepts and redirect links along with their categories. I then analyze the extracted information and form term-concepts, concepts and categories vectors. The class diagram shown in Figure 5.2 illustrates the main components performing these stages. The following sections further describe these stages and elements.

5.2.1 Preprocessing the Wikipedia Dump

I used a snapshot of Wikipedia that was formed in 16-03-2010. The data was provided in an XML file prepared in different packages (different languages, containing only page titles, edit history, etc.). I chose the English package named *pages-articles.xml* and performed several operations to prepare it for the next stage before it is analyzed.

- First I removed the non-relevant fields in each article such as edit history. I also removed all non-English terms contained within each Article.
- Then, I parsed the Wikipedia [Templates] format and resolved the links. I counted the number of links existing within each article.
- Afterwards, the text of each article was segmented into words.
- Articles that are too short containing less than 100 words or fewer than 5 links are removed. Articles that are too short or incomplete are called *stubs* in Wikipedia. In this step it is ensured that they are removed in addition to any others having high likelihood of being stubs. This was mainly to increase the reliability of the system by removing articles which are deemed too short to provide encyclopaedic coverage of a subject. Also, if the title of an article contains only one term which happens to be in the stop word list, I remove that article.
- Articles that belong to categories related to chronology such as Years and Centuries are removed.

The total number pages in the original dump before processing is 3,289,927. After applying the above-mentioned rules, I had a total of 1,504,748 articles where each article represents a unique concept. The total number of categories I had was 126,709. Parsing the main XML file that stored Wikipedia's contents was performed by the class *WikiXMLParser* shown in Figure 5.2. The class *WikiPageIterator* iterates through all the pages stored which are parsed individually by *WikiPageParser*, filtered as needed by if belonging to the excluded Categories types, and cleaned by *ArticleCleaner* to remove non important tags, non English characters, non relevant data such as tables and Infoboxes and extra spaces that may appear in the titles. Each parsed page is represented by the class *WikiPage* which stores all of the page data in different variables after applying some analysis to its contents through its member functions. Categories and links are also represented by the classes *Category* and *link*, respectively. More details about the rest of the classes are presented within the description of their corresponding stages in the next subsections.

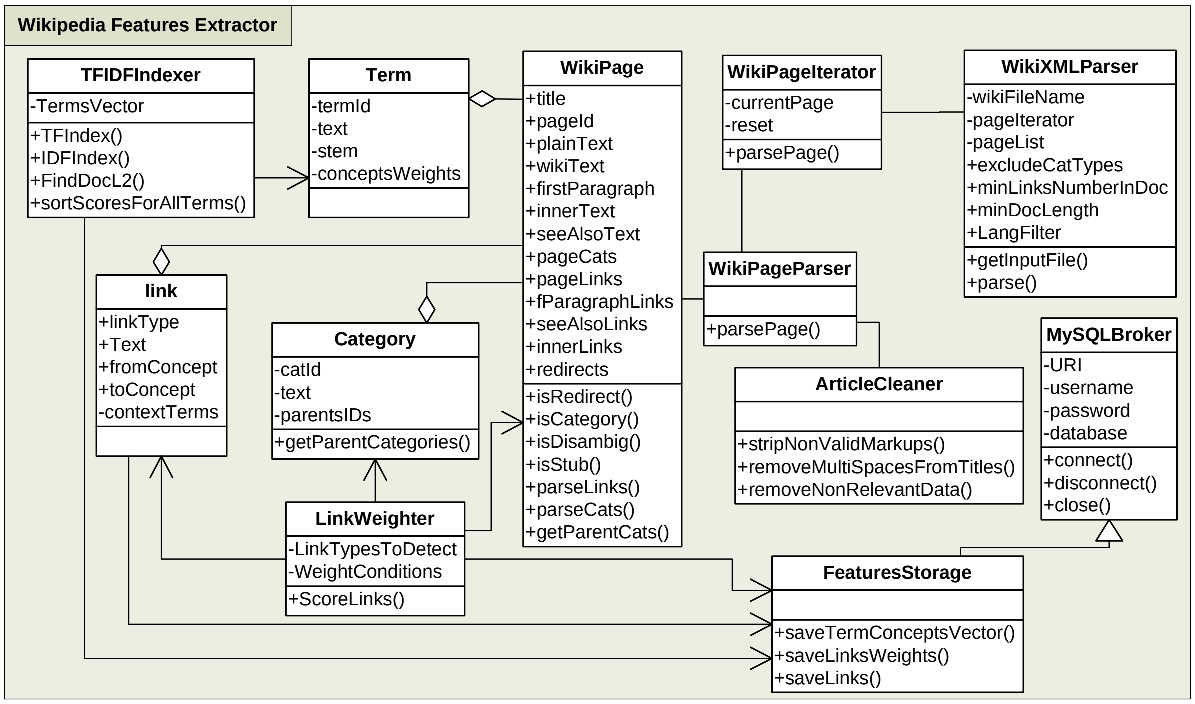


Figure 5.2: Class Diagram showing the main classes of the *Wikipedia Features Extractor* package

5.2.2 Extracting the Features from Wikipedia

The preprocessed version of Wikipedia is analyzed to leverage the articles contents, their titles, redirect links and the categories structure. First, I removed all stop-words from the articles content. The remainder terms were used to serve the purpose of representing all of Wikipedia Concepts. This was achieved by examining the terms distributions within each article and computing the weight for each word in the form of TFIDF which is one of the most common weighting methods used to describe documents in the vector space model. TFIDF factors two aspects for each term: its frequency within each document (represented as TF) where the higher the TF in a document the more chance that it is important within that document. The second factor is the Inverse Document Frequency IDF where a word is deemed more important in a document if it doesn't appear in many of the test collection documents as was described in section 2.3.1.1.

5.2.2.1 Term-Concepts Table

In essence, I map all the terms existing within each article to all the Wikipedia Concepts creating a vector for each term whose elements are the l2-normalized term weights within each Concept. These weights resemble how much the terms contribute to each Concept they are attached to. I rank the concepts each term belongs to based on the formed weights in a decreasing order to form the Term-Concepts table. The top concepts in the list are the most relevant ones to the term. For example, the term *Birmingham* has the following associated concepts *history of Birmingham*, *Birmingham* (the English City), *Birmingham Alabama* (the American City), *Arts in Birmingham*, *Timeline of Birmingham History*, *Birmingham City University*, *Barry Vincent Jackson*, *B Postcode Area*, *Birmingham Local Elections, Economy of Birmingham*, *Birmingham Business Journal, etc.*. One can notice that the covered range of different Concepts varies from city name (in UK and USA), to events that occurred in one of the two cities, to a person name who owned a theatre in the English Birmingham.

As mentioned above, I rely on terms distributions within each Wikipedia article to determine the weight of each term in relation to each concept. One can view the resulted structure in the form of a non-negative weight table or a sparse matrix where rows correspond to terms and columns correspond to concepts. The weights are computed by using the common TFIDF metric as follows:

$$Weight^{'}(t,c) = tf(t,c) \cdot \log(\frac{n}{df(t)}) \qquad \textbf{5.1}$$

Where *tf(t,c)* is the term frequency of the term *t* in the article (or concept) *c*, *n* is the total number of articles in the evaluation set, and *df(t)* is the number of articles containing the term *t*. The weight is then l2-normalized to account for the different lengths in Wikipedia articles by applying the following:

$$Weight(t,c) = \frac{Weight'(t,c)}{\sqrt{\sum Weight'(t,c)^2}} \qquad \textbf{5.2}$$

The *TFIDFIndexer* class shown in Figure 5.2 is responsible for building the Term-Concepts table in the implemented system. After building the table, I perform a two-level update to it through a series of iterations focusing on boosting the weight scores for some terms based on their appearances within the titles and redirects links. This is explained further in the following section.

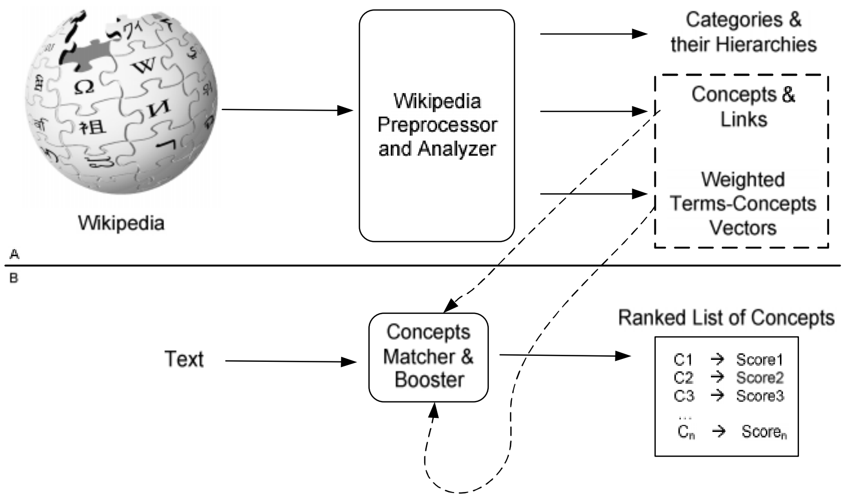

Wikipedia
Wikipedia Preprocessor and Analyzer
Categories & their Hierarchies
Concepts & Links
Weighted Terms-Concepts Vectors
A
B
Text
Concepts Matcher & Booster
Ranked List of Concepts
C1 → Score1
C2 → Score2
C3 → Score3
...
Cn → Scoren

Figure 5.3: an Overview for how the Concepts Matcher & Booster produces the Ranked Concepts. Part A of the figure is performed only once while part B is an integral part of the system that is repeated every time a new text fragment is processed.

5.2.2.2 Concepts-Boosting

In the *Birmingham* term example mentioned above, I notice the occurrence of the term *Birmingham* in many of the Concepts titles or the text of the articles they belong to. This is not always the case for some other generated concepts. Take the word *Unhappy* as an example. Some of the concepts the word is related to do not have any occurrence of the word *Unhappy* in their titles or even the Wikipedia article text they represent. For example, the concept *Depression (mood*) does not have any reference to the word *unhappy* in its text or its title, yet it is related to it. With the sole help of the Term-Concept table previously built, the Concept *Depression (mood)* would not appear in the list of related concepts for the term *Unhappy* because the TF for that term in the concept's article text is zero. In a similar way, some concepts titles may contain the keyword *Unhappy* in their titles which should give them a higher tendency to be more related to the term than many other concepts. For instance, the concepts *Unhappy Consciousness*, *Unhappy Triad* and *Unhappy Happiness* are assumed to be more related to the term *Unhappy* due to them all sharing the key term *Unhappy*.

To tackle the above-mentioned issues, I thereby apply a two-level *Boosting* process as a following step after generating the concepts vector using the term-concepts table for any given term. In essence, I make use of the large number of Redirect links existing within the Wikipedia structure by analyzing the keywords of the title of each Redirect link in addition to the titles of the articles they link to. In the first boosting level for a term or a group of terms ***w***, I hypothesize that a redirect link ***r*** containing only ***w*** in its title should link to a concept ***c*** that is highly related to ***w*** regardless of whether ***c*** has ***w*** in its article content. In other words, ***w*** and ***c*** should have a high relatedness score which is achieved through the boosting performed in my algorithm by assigning a score to ***c*** based on the value of a variable I call *FirstLevelBoost* that was previously determined. In a similar way, I apply the same idea to the Concepts Title $\boldsymbol{c_t}$ and word ***w*** to generate a relatedness score for ***c*** using the same variable *FirstLevelBoost*. To generate a value for the $FirstBoostLevel_w$ of a term *w* after the above-mentioned conditions are met, I apply the following formula.

$$FirstLevelBoost_w = \underset{c_s \in C}{Max}(c_s) \cdot FirstLevelBoost \qquad \textbf{5.3}$$

Where C is the set of concepts found related to the term w, c_s is a concept score as computed in the term-concepts table and *FirstLevelBoost* designates how much boost is applied. The optimal value I found for this based on my expirments is 1 which effectivelys renders the matched concepts in the top of the term-concepts list.

In the second boosting level, I examine the occurrence of the term or group of terms ***w*** in the Concept Title $\boldsymbol{c_t}$ or the redirect link ***r*** that points to it. If $\boldsymbol{c_t}$ or ***r*** contains ***w*** in addition to some other terms, I increment the relatedness score of ***c*** by a value based on the previously chosen variable *SecondLevelBoost*. The resulted relatedness score from the second level boost will always be less than the first level boost. Also, as the number of terms appearing within the title or redirect link increases, the amount of boost being applied inversely decreases. This is reflected in the following formula being applied to generate a value for the *SecondLevelBoost* of the term *w*.

$$SecondLevelBoost_w = ((SecondLevelBoost - 1)^{SecondLevelBoostAdj} + 1) \qquad \textbf{5.4}$$

The value of *SecondLevelBoostAdj* in the above formula is computed as follows:

$$SecondLevelBoostAdj = \frac{|c_t|}{N} \qquad \textbf{5.5}$$

Where | $\boldsymbol{c_t}$ | is the number of terms that exist within the concept title (or the redirect link), and N is the number of times the term w appears within the concept title.

The algorithm I apply for both boosting levels is illustrated in Figure 5.4 where W refers to the set of words that I require the most related concepts to, C is the set (c_t **,** c_s) for the concepts that resulted from applying the Term-Concept table method to W, *allC* is all the Concepts within Wikipedia and *allR* is all the redirect links. The result of applying both boost levels on the two terms *Unhappy* and *Jobless* is shown on Table 5.1 while the pseudo code for the concepts boosting algorithm is illustrated in Figure 5.4.

	Unhappy	***Unhappy* (Boosted)**	***Jobless***	***Jobless* (Boosted)**
1	Implications of Divorce	Depression (mood)	Growth Recession	Unemployment
2	Unhappy Consciousness	Unhappy Consciousness	When Work Disappears	Jobless Recovery
3	The Better Half (play)	Happy Number	Pôle Emploi	James Renshaw Cox
4	The Human Contract	Unhappy Triad	James Renshaw Cox	Growth Recession
5	Kurumi Enomoto	Fan the Flame (part 1)	Joe Ma Wai-ho	When Work Disappears
6	Pamela Springsteen	Unhappy Happiness	Vetti	Pôle Emploi
7	Tristan Davies	Implications of Divorce	Volksgrenadier	Joe Ma Wai-ho
8	Fan the Flame (part 1)	the Better Half (play)	shadowstats.com	Vetti
9	Notes & Rhymes	the Human Contract	Jobless Recovery	Volksgrenadier
10	Ballad of a Teenage Queen	Kurumi Enomoto	Imperfect Competition	shadowstats.com

Table 5.1: **Boosting the Term-Concepts Vectors using Redirect Links**

```
Boost_Concepts( W, C, FirstLevelBoost, SecondLevelBoost, allC, allR){
            HS ← Max( $c_s$ ∈ C) /* Highest Score in the C set */
            FL ← FindFirstLevelConcepts(W, allC, allR)
            FLB ← ϕ
            ∀ ($c_t$ , $c_s$) ∈ FL, FLB ← FLB ∪ ($c_t$ , (HS * FirstLevelBoost) )
            SL ← FindSecondLevelConcepts(W,C, SecondLevelBoost)
            Res ← C
            Res ← Res ∪ FLB ∪ SL /* Overwriting repetitions with those in SL and FLB */
            return Res
}

FindFirstLevelConcepts(W, allC, allR) {
   Res ← ϕ
   Loop over ($c_t$ , $c_s$) ∈ (allC ∪ allR) {
     /* all elements of W are contained within $c_t$ */
    If ((W ≡ $c_t$) and (len(W)= len($c_t$)) Then Res ← Res ∪ ($c_t$ , $c_s$)
   }
   return Res
}

FindSecondLevelConcepts(W,C) {
   Res ← ϕ
   Loop over ($c_t$ , $c_s$) ∈ C {
     If ( (∀ w ∈ W) ∃ c' where (c' ≡ w) and (c' ∈ $c_t$) and (len(W) <> len($c_t$)) ) Then
       $c_s$ = $c_s$ * ( ((SecondLevelBoost-1)^(|w| / | $c_t$ |))+1)
       Res ← Res ∪ ($c_t$ , $c_s$)
     EndIf
  }
   return Res
}
```

Figure 5.4: **the Pseudo Code for the Concepts-boosting Process and its Subroutines.**

5.2.2.3 Wikipedia Links and Categories Structure

Wikipedia can be viewed as a semi-structured encyclopaedic resource. It is not as well structured as a database for example. It has pages containing text segments of varying size and format. However, it also has semi-structured pages for ambiguous terms listing their possible meanings with links to the articles describing them. It has structured categories attached to each article. The categories have parents and children relationships defined among them. Articles belonging to the same category have generally similar outline and structure. In addition, over 86 million links exist within it linking articles with each other. These links are of different types and can be a representative for some form of relationship between articles with each other. The categories too with their hierarchy can help better enhance the definition of the semantic relationship between the articles.

In this section, I focus on using the links and categories structure in Wikipedia to enhance the features I previously extracted. Wikipedia contains different types of links. There are interlanguage links linking to versions of the article in different languages. There are internal links linking to other pages within the same Wikipedia language. There are interwiki links pointing to other pages within the projects hosted by Wikimedia but not necessarily to Wikipedia articles. There are also external links to pages outside the Wikimedia-operated projects. Many other types of links exist too within the articles such as section links, date links, and template links. My focus here is on the internal hyperlinks in the articles text pointing to different English articles in Wikipedia.

Not all the internal links are of the same significance. Some links may be more reflective of the relatedness of an article to another than many others. To illustrate this, take the two links *Basketball court* and *Peripheral Vision* existing within the article about the famous *Basketball* sport game as an example. Intuitively, the former link is more related to the article than the second. It is thus important to apply some form of filtering to the article links to reduce the resident noise and embrace those that link to most related articles. Therefore, I devised my own links-filtering module. The module's goal is twofold. First, it reduces the number of noisy or unimportant links by focusing only on potentially high-quality links. Second, it enhances the overall efficiency of the system since the total number of links to be evaluated and analyzed will be reduced. This is especially evident for those articles that contain a large number of incoming links such as the article about the famous company *Google* which has over 70,000 incoming links. The analysis of such large number of links for all the articles in Wikipedia would require a large amount of computing resources and is simply not efficient.

In the filtering module, I classify the internal links into several levels signifying their importance based on my own observations. As for the categories, I first attempted to utilize the category structure within Wikipedia directly on its own but realized that even though some categories are narrow and indicate strong relatedness between their articles, some other categories are broad and not as useful as many others. For example, the category *Historiography* is broad and has 129 pages. Among these pages are *Silver Age* and *Source Text* which can not be said to be strongly related to each other. Due to the generality of some categories and because I still think that categorization can be useful especially for some narrative categories although to a limited extent, I chose to filter the categories I use with the internal links. Figure 5.5 shows the links types I defined sorted based on the weights they carry in a decreasing order. In general, it is possible to divide the defined link types into three categories: Mutual Links where two articles directly link to each other, One Link with shared a Parent Category, and See Also links which are usually appended to most of the articles in Wikipedia. I next define and describe these link types along with the weight level I assigned to each:

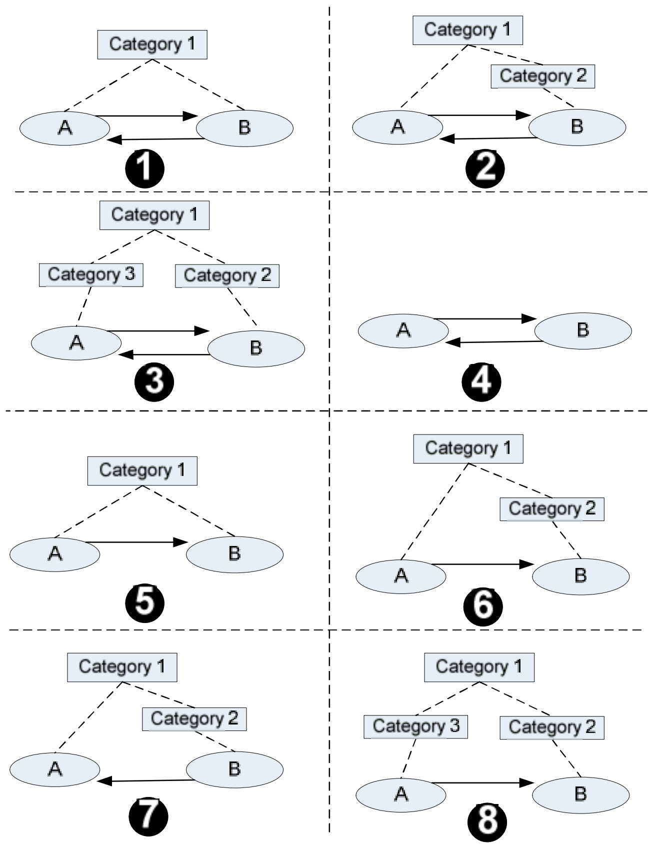


Figure 5.5: Link Types defined sorted based on their weights in descending order

w_1	3	**w_2**	2.75	**w_3**	2.5	**w_4**	2.25	**w_5**	1.75
w_6	1.5	**w_7**	1.5	**w_8**	1.25	**w_9**	3.75	**w_{10}**	3.25

Table 5.2: Weights assigned for the different links types

Mutual Linking

When article A contains a link or more pointing directly to article B, and article B contains a link or more pointing to article A, I consider these links for the two articles to be of a high value signifying a strong relatedness between the two articles. An example for this is the *Basketball* and *Slam dunk* articles which both contain links pointing to each other and they are closely related. If the two articles share one or more parent categories, they are expected to be much more related than if they were not. Thus, I classified the mutual link types into four classes 1-4 as shown in Figure 5.5. In 1, both articles directly share a parent category. In 2, the parent category of one article is a subcategory of the parent category of the second article. In 3, both articles share a grandparent category that is exactly one category-level away from the articles. In other words, the articles belong to at least two categories whose parents are the same. In 4, no shared parent or grandparent category is found and thus only the reciprocal links are considered. All of the four link types 1-4 resemble strong relatedness between the two articles in each case when compared with the types 5-8 but of varying weight. The weights I assigned to each link type are illustrated in Table 5.2.

Shared Parent-Category with One Link

As highlighted above, merely having a shared category between two articles may indicate a strong relevancy between the articles. However, this can not be applied as a general rule due to the breadth and generality of some categories. Therefore, I adopted the "*at least*" one link sharing rule as a filtering mechanism. I also expanded it to include grandparent categories in some cases, namely link types 6, 7 and 8 in Figure 5.5. As a general rule for the link types covered in this category of links, I say that when Article A points to Article B or B links to A AND both articles belong to the same category (or grandparent category), I have a potential strong relevancy between the two articles A and B. An example for this category of links is the articles titled *Great Depression* and *Panic of 1893* which both belong to the category *Financial Crisis* and the former article has a link pointing to the second. Both articles discuss the economic depressions that occurred before the Second World War. However, Until the *Great Depression*, *Panic of 1893* was considered the harshest depression in the history of the United States. The relevancy between these two articles is thus greater than the relevancy between either and the rest that only share one parent category with either of them such as the articles *Bad Bank* and *Bank Run* both under *Financial Crisis* category.

See Also Links

These links are usually added manually by Wikipedia volunteer editors to the end of the articles in Wikipedia and refer the readers to other semantically related topics to the current article. I give a high weight to these types of links and label them with w_9 in Table 5.2. I also give a high weight to the inverse of the See Also links labelling them with w_{10} in Table 5.2.

5.2.2.4 The Rationale behind Choosing the Links and their Weights

The idea of using the links within Wikipedia article was inspired by my study on the links themselves and the use of websites links by search engines. I observed that in many cases, there is some relatedness between an article and the articles being pointed to by the anchors in that article. Just like the idea used by many search engines, links from pages in sites of similar context pointing to a specific website indicate some importance for that website which is being linked to. Moreover, a relatedness can be induced between all of the websites given that they all share similar contexts. In my work here, I investigate the links present within the Wikipedia articles, classify them, give them weight, and deduce the relatedness between the target article and the anchor text articles.

After manually assessing the links presented in many articles, I observed that the anchors presented in one article can be classified into two categories: reciprocal links and one-way link. This is also evident in the information retrieval domain with many websites forming mutual links to point to each other. For example, a website specializing in selling mobile phones may be pointing to another website selling mobile phone accessories, or they can both be linking to each other. Sometimes, both websites fall under the same category, "Mobile Phone" for the previous example. My method applies a similar concept to Wikipedia by using the Categories structure and links for defining a relatedness measure between the articles. The weights I assign to the different links resemble the context-ranking method applied by search engines when evaluating the internal and external links in a webpage to combat and penalize Links Farming[12]. The context filtering and selection in my method does not match the same complexity as those methods implemented by search engines and it is actually not needed since Links Farming is not necessarily present in Wikipedia. However, the general goal is the same: rank pages or articles based on their relatedness to the target page.

The choices of the different links weights reflect the importance level of each link type. It is intuitive to say that articles with mutual links are more related than articles sharing single incoming or outgoing link. To demonstrate the effects of the different weights assigned to the different link types and illustrate how they were chosen, I apply the extracted features in the application of WSD. In the evaluation, I analyze how the weight of each link affects the performance of the system and report the results I obtain. The results I obtained are provided in section 5.4.4.

In [165], a methodology was used for detecting the explicitly-mentioned Wikipedia concepts within documents. The authors applied that methodology in the task of documents clustering. With the method proposed here, I also consider the related concepts to those explicitly mentioned in the text documents and assign different weights to each depending on how strongly related they are to the Exact-Match concepts. The effect of adding the different weights is shown in the results illustrated in section 5.4.4. In that same figure, I also show the performance when considering the links without any weight assignments (Unweighted strong links).

[12] A Link Farm is a group of websites that all link to each other for the purpose of increasing their page rank. There doesn't necessarily exist a context relationship between the sites in the farm.

5.3 Examples of Wikipedia-based Features Generation

I provide here examples for how features are extracted using both the term-concepts table and Strong Links methods. For each given text, I show only the top 10 most related concepts using both methods due to the limited space available. For some concepts, I provide brief descriptions taken from the corresponding Wikipedia articles themselves. The examples are illustrated in Table 5.3.

Text	Term-Concepts Table Features	Strong-Links Features
Cable News Network	CNN Controversies	United States Cable News
	CNN International	CNN Warner
	CNN News Stand	Ted Turner
	CNN Center	CNN Center
	CNN.com Live	Time Warner
	CNN Pipeline	Time Warner Center
	Melissa Long – CNN news anchor and reporter	Lary King Live
	CNN Sports Illustrated	Media Bias
	Talkback Live – talk show on CNN	Media Matters for America
	United States of Cable	RealNetworks
Google purchased Youtube one year and a half after its launch	Outline of Google	Google
	Google Business Solutions	Youtube Live
	History of Google	Fair Use
	Google	Viral Video
	Censorship by Google	CBS
	Youtube	Adobe Flash Player
	Youtube Censorship	High Definition Video
	Criticism of Youtube	Paypal – Youtube was founded by 3 former Paypal employees
	History of Youtube	Alternative Media
	Google Apps	1080p

Table 5.3: Examples of Wikipedia-based Features Generation

The second example in the table included more than one concept in a sentence. For text fragments that contain more than one concept, the aggregated list of concepts takes this factor into account and disambiguation is usually required. In the next section, I describe how the features have been applied to the application of Word Sense Disambiguation and how concepts aggregations within text fragments or sentences take place.

5.4 Word Sense Disambiguation

The task of WSD is to automatically predict the right sense for a specific term in the given context. In my system, I use the local features presented in the given context along with the previously-extracted Wikipedia features to achieve this task. Figure 5.6 gives an overview of the modular design in my WSD system. In each instance run, a text document is fed to the system which can be a text fragment, a sentence, a paragraph or a whole document. In the text document, a single term is marked as the target word to be disambiguated, and is displayed as a separate input in Figure 5.6 for illustration. The provided document is then processed sequentially in each module in a pipeline and the final output of the system is the target sense for the marked word. In essence, the system predicts the correct concept (or sense) for the target term by applying the term-concepts table to the target term and then scoring each concept according to my analysis of the strong links. The result is a ranked list of scored concepts with the top concept in the list being the most likely sense for the target term and is produced as the final result. In the following subsections, each module in the WSD system is described.

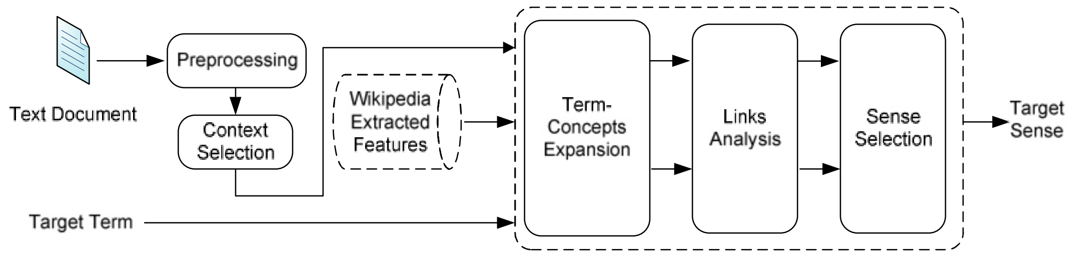


Figure 5.6: **an Overview of the WSD process**

5.4.1 Preprocessing and Context Selection:

Based on the format of the input text document, its content is parsed first and its terms are extracted. Stop words are then removed from the document. This is followed by a preprocessing step in which the marked target term is highlighted to determine its context based on its surrounding words. If the supplied text document is in the form of a short text fragment (less than a predefined number of words), all of its surrounding terms are considered. Otherwise, I consider extracting 2n words surrounding the target term, n words before and n words after, which I call Context Terms (CT) in the following steps. I can label each CT with $ct_{i=1\ldots|CT|}$ where i is the context term number in the generated list.

5.4.2 Term-Concepts Expansion

After obtaining CT from the Context Selection module, the term-concepts vector is applied on the resulting CT and also the target term. The number of concepts lists that would be generated is |CT| + 1. If I label each concept list with C $_{i=1\ldots|CT|}$, I would have the concepts list defined as:

$$C_i = \{c_{ij}\}_{j=\{1\ldots V\},i=\{1\ldots|CT|\}} \qquad \textbf{5.6}$$

Where i is the number identifying the C_i concept list, j is the concept number in the list and V is the total number of c_{ij} concepts in the concept list. As for the target term concepts list TW, I define it as:

$$TW = \{g_k\}_{k=\{1\ldots M\}} \qquad \textbf{5.7}$$

Where g_k is the concept numbered k in the target term concepts list and M is the number of concepts in the list.

5.4.3 Links Analysis and Sense Selection

In the built system, I use the local features presented in the given context along with the features previously extracted from Wikipedia to detect the sense of an ambiguous term. The links contained within Wikipedia articles are among the features I examined. I illustrate in this section how they can be used to determine the relatedness between two concepts and aid in the problem of WSD. I present different methods employing different aspects of the extracted features and compare their performance in the evaluation section.

When analyzing the internal links present in an article, I consider only those that fall into the category of one of the above-mentioned types. I use the links as part of a process to compute the relatedness between two articles A and B. This is achieved in my work by two methods: (i) directly examining the links present in both A and B and computing a score for each link. (ii) forming two sets of articles S_A and S_B where each set would contain the most relevant articles to A and B respectively and then computing the similarity between the two sets using the Cosine distance. The following subsections describe how the methods are implemented in the application of WSD.

5.4.3.1 Terms Vectors Intersection

In this basic method, I examine the CT surrounding the Target Term first and form a term vector called T containing both CT and the Target Term. This can be represented as:

$$T = \{w_y\}_{y=\{1......|CT|+1\}} \qquad \textbf{5.8}$$

where y identifies each term in the list. Suppose the Target Term has multiple meanings represented as S_{ye}, where y is the Target Word number in the list T and e is the meaning (or sense) number. Since each sense is associated with an article in Wikipedia, it is possible to attach each sense with the most dominant words in the document it represents using the TFIDF measure. After all, this was already precomputed when forming the

Term-concepts table. It can be chosen as the top R terms with the highest TFIDF scores. The list of dominant terms for each sense can be represented as:

$$S_{ye} = \{a_{yef}\}_{f=\{1...D\}} \quad \textbf{5.9}$$

Where f is the number of the word a in the dominant words list of S_{ye}. For each candidate sense e, I compare its dominant words list S_{ye} with T. The candidate that would be chosen as the representative for the Target Word is the one having the majority of its terms in S_{ye} present in T.

To illustrate the above method, I give the following example. Suppose that a document contains the following T list:

T={ group, Foxtel, *Fox*, CBS, international, team}

The underlined word is the Target Word which I aim to find the right sense for. Suppose that it has two possible meanings: (1) Fox (animal) and (2) Fox Broadcasting Company. Each meaning has an associated article in Wikipedia and thus the dominant words in each article as computed by the TFIDF measure can be represented in a simplistic view as:

S_{31} = { Mammals, tail, dog}

S_{32}= { Foxtel, entertainment, CBS}

Now, an intersection of S_{31} and S_{32} with T yields the following:

$S_{31} \cap T = 0$
$S_{32} \cap T = 2$

This shows that the second sense S_{32} (Fox Broadcasting company) is the most representative for the Target Word in the given document.

5.4.3.2 Unweighted Strong Links

In this method, the *CT* surrounding the Target Term are also examined first and a term vector called T containing both *CT* and the Target Term is created. The Target Term would have multiple meanings, and each meaning is associated with a Wikipedia article. I compile a list of related articles for each meaning based on the strong links analysis performed in section 5.2.2.3. It should be noted that the links weights are not considered here. The list of most related articles (or concepts) for each meaning as devised from the strong links is represented by the following formula:

$$P_{ye} = \{a_{yef}\}_{f=\{1...Q\}} \tag{5.10}$$

Where f is the number of the related concept to the concept ye, e is the meaning number and Q is the maximum number of related articles according to the strong links analysis. Just like in the above method, for each candidate meaning, I compare its most related articles with T. The candidate that would be chosen as the representative for the Target Word is the one having the majority of its terms in P_{ye} present in T.

I present the following example to illustrate the described method. Suppose that a document contains the following T list:

T={ group, Foxtel, *Fox*, CBS, international, team}

Just like in the previous example, the underlined word is the Target Word which I aim to find the right sense for. It also has two possible meanings (1) Fox (Animal) and (2) Fox Broadcasting Company. Each meaning has a list of most related articles represented in the following vectors:

P_{31} = { Animal, Species, tail, dog, Popular culture }

P_{32}= { Foxtel, United States, News Corporation, CBS}

Performing the intersection here again of P_{31} and P_{32} with T yields

$$S_{31} \cap T = 0$$
$$S_{32} \cap T = 2$$

And this also shows again that the second sense is the best candidate for the Target Word Fox in the given context.

5.4.3.3 Weighted Strong Links

In this method, I analyze the links present in the article content of each concept in C_i against all concepts in *TW*. I propose performing the comparison between any two articles indirectly through expanding the two articles into two lists and then comparing the two lists with each other. This can be translated for my WSD process into the following:

$$eTW = \{G_k\}_{k=\{1...M\}} \quad \textbf{5.11}$$

eTW is the expanded list for *TW*. It contains a list G_k of related articles for each possible sense g_k. The list of G_k is formed as follows:

$$G_k = \{(gc_w, f(g_k, gc_w))\}_{w=\{1...V\}, f(g_k, gc_w)>0} \quad \textbf{5.12}$$

In the above formula, gc_w is any article that is related to g_k. The function $f(g_k,gc_w)$ measures the relatedness between the two concepts g_k and gc_w based on the link analysis and the weight assigned to each link. *V* refers to the total number of available concepts for the set G_k after the expansion. I apply the same process to all the concepts existing within C_i and then group them all together in one set summing the score for repeated concepts if they occur in more than one set. I thus obtain the following as a result of the later expansion of C_i:

$$eC_i = \{(rc_{ijw}, f(c_{ij}, rc_{ijw}))\}_{j=\{1...V\}, i=\{1...|CT|\}, w=\{1...Q\}, f(c_{ij}, rc_{ijw})>0} \quad \textbf{5.13}$$

Where *w* refers to the number of the related concept rc_{ijw}, rc_{ijw} is the concept related to c_{ij}, and *Q* is the total number of related concepts. The resulted score $f(\ c_{ii}\ ,\ \ rc_{ijw}\ )$ can be written as tw_{ijw} for abbreviation. I sum all of the resulting related concepts eC_i into one list *eC* where repeated concepts rc_{ijw} are grouped into one by summing their weight:

$$eC = \{(rc_v, tw_v)\}_{v=\{1...|D|\}} \quad \textbf{5.14}$$

$$tw_v = \sum_{w=1}^{Q} (tw_{ijw}) \text{ , where } rc_w = rw_v \qquad \textbf{5.15}$$

Where rc_v is a unique concept in eC_i, D is the total number of *unique* concepts in eCj, tw_v is the weight given (after computing the sum) for concept rc_v.

After obtaining the list *eC* of related articles derived from the context terms, and a list G_k for each meaning of the target term, I compute the distance between *eC* and each G_k using the cosine distance measure. The final answer for the WSD problem would be the concept carrying the number *k* where:

$$\max_k (dist(G_k, eC)) \qquad \textbf{5.16}$$

I examine the same example used with the previous method to illustrate this method.

Target term = Fox

Doc = {Foxtel, <u>*Fox*</u>, CBS }

Foxtel → C_1= { Foxtel_{11}, $\text{Optus Television}_{12}$} = { c_{11}, c_{12} }

CBS→ C_2= { CBS_{21} , $\text{The Early Show}_{22}$ } = { c_{21} , c_{22} }

Fox → TW = { Fox (animal)_1 , $\text{Fox Broadcasting Company}_2$} = { g_1, g_2}

When applying this method to the given example, C_1 and C_2 are expanded into the following (applying Equation 5.12):

eC_1 = { (Optus Television, 3) , (HDTV, 2.5) }

eC_2 = { (20th Century Fox , 6.75) , (NBC, 3) }

Which are then grouped into the following:

eC = { (Optus Television, 3) , (HDTV, 2.5) , (20th Century Fox , 6.75) , (NBC, 3) }

TW on the other hand is transformed into the two weighted lists:

G_1 = { (silver fox, 1.75) , (Pierson v. Post,1.5) }

G_2 = { (20th Century Fox , 3), {NBC, 3) }

And thus, when I compute the relatedness between *eC* and G_1 using the cosine measure I get 0.0 while getting 0.4 for *eC* and G_2. Therefore, I conclude that G_2 is the most likely sense for the given context.

Even though the given example above is very simple and only shows a small portion of the concepts extracted for each term in *TW* and Doc, it is worth noting that the actual number of possible senses (or concepts) given for each word is much larger and more diverse. This is mainly due to the expansion of the terms using the boosted term-concepts table described earlier. This can be seen as an advantage to the system when compared against many others that attempt to use the disambiguation pages in Wikipedia to define the possible senses of a term such as [166] and [167]. In many cases, the provided disambiguation pages are not comprehensive and may require manual intervention to match many concepts in the page against the main title of that page. For example, the disambiguation page for the term *Apple* lists has the following as a possible meaning for the term Apple:

Apple Martin, the daughter of *Gwyneth Paltrow* and *Chris Martin*[13]

In the above line, Apple Martin is the referred character and is not hyperlinked. However, only Gwyneth Paltrow and Chris Martin are hyperlinked. Without manual analysis, an automatic system would most likely choose wrongly one of the two hyperlinked characters as the possible meaning for the term Apple. To improve the overall efficiency of the system while running the evaluation experiment, I applied some filtering to limit the number of concepts produced from the link-based expansion. This is further explained in the evaluation section.

[13] Words in Italic are hyperlinked in Wikipedia

5.4.4 WSD Evaluation

The datasets being used for evaluating WSD systems greatly depend on the variations of the parameters for systems being evaluated. Senseval-1/2/3 and SemEval test collections are based on WordNet making them difficult to use directly with my methods. This is mainly in part due to the differences in the words senses defined in WordNet when compared with those available in Wikipedia. Mapping senses from WordNet to Wikipedia has proved to be a difficult task [168] and requires evaluation itself. In addition, due to having WordNet developed by linguistic experts, common words happen to have a large number of senses with small differences between them [169].

Therefore, I created my own benchmark which is most similar to those devised in the recent work of Turdakov [170] as well as others in [171] and [172]. I use the manually created links in Wikipedia as the basis for my dataset. The internal links of Wikipedia look in the following form: [[Part1 | Part2]] where Part1 is the text in the hyperlink or the title of the page being linked to while Part2 is the text displayed to the reader when reading the article containing this link. An example for this is the text fragment "With colors such as shades of [[brown (color) | brown]]" in the article titled *Rabbits*. Clicking the word *brown* in that fragment redirects the reader to the article *brown (color)*.

I chose the mentioned technique to construct a dataset of 1,000 anchors, along with their correct meanings and the paragraphs that contain them, which were selected from 100 random articles from Wikipedia. During the evaluations, I chose 20 words surrounding the target word TW as the context terms. This choice was based on an experiment I performed on the third method for which the results are shown in Figure 5.9.

In an effort to choose the most optimal weights for the strong links, I performed an experiment in which the weight of one link is changed while keeping the rest unchanged.

The effect of the change is reflected in the performance of the system in the WSD task and is shown in Figure 5.7 for weights w_1 to w_5 and in Figure 5.8 for w_6 to w_{10}. It can be noted from the results that the See Also and Inverse See Also links have the strongest effect on the system as the accuracy degrades to 65.91% and 66.41% respectively when their weights are set to zero. A link between two articles sharing a grand parent category is found to be the weakest of all as the accuracy of the system is decreased to 75.31 when the link weight w_8 is set to zero. Setting w_8 to any value above 1.25 negates the performance, too. The effect of w_6 and w_7 is very similar and their curves on the chart in Figure 5.8 almost match. In general, representative lines for mutual links (w_1-w_4) show good performance when the links weights are set above 2.5. This is not the case with single links as the system performance degrades when the weights are set above 1.75.

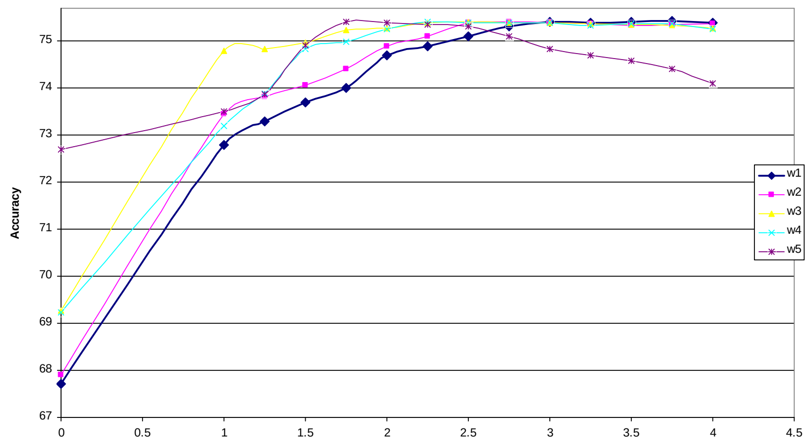


Figure 5.7: Effects of Strong Links Weight Change for w_1 to w_5

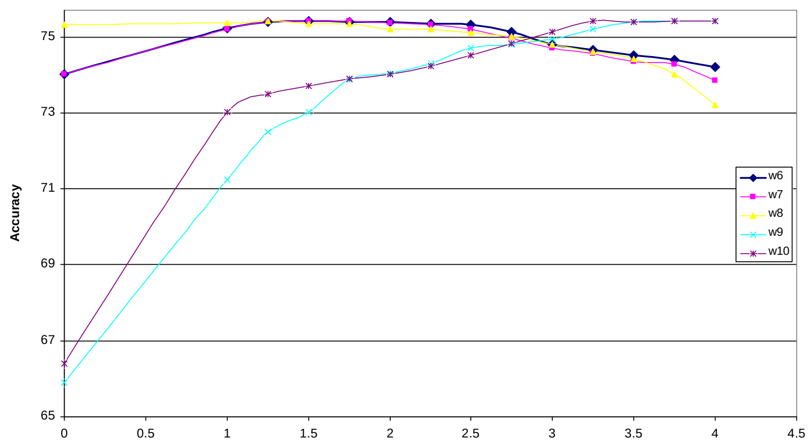


Figure 5.8: Effects of Strong Links Weight Change for w_6 to w_{10}

I performed several experiments on the built dataset to test my methods and the effect of introducing the chosen links: (1) Term vectors Intersection, (2) Unweighted strong links, and (3) Weighted strong links. The best result I obtained was with weighted strong links method giving an accuracy of 75.41%. This shows that including the strong links for analysis does indeed improve the overall accuracy of the system especially when compared with the Term vectors intersection (69.17%). The accuracies obtained for all the methods during the evaluations are shown in Table 5.4. Since I produce a ranked list of weighted potential meanings in each method, I considered computing the chances of having the right sense in the top-2 and 3 senses of the produced list. The results I obtained show that the accuracy of the best method increases to 91.82% when considering the top-3 senses in the list.

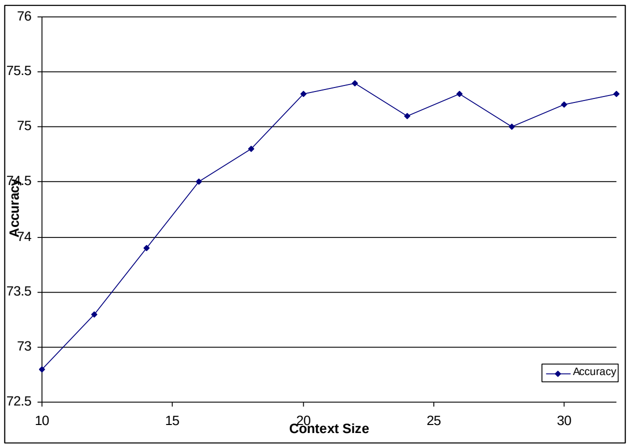


Figure 5.9: Effect of Context Size on the Accuracy in WSD with the Simple Link Analysis method

	Top	**Top-2**	**Top-3**
Term Vectors Intersection	69.17	75.8	82.71
Unweighted Strong Links	71.84	84.08	87.29
Weighted Strong Links	**75.41**	**87.19**	**91.82**

Table 5.4: The accuracy of all implemented methods including (i) Term Vectors Intersection, (ii) Unweighted Strong Links, and (iii) Weighted Strong Links methods.

5.4.5 Conclusion

In the previous subsections, I described how to apply the Wikipedia-extracted features, namely the term-concepts table and strong links, to the application of WSD. I used the manually created links in Wikipedia as the basis for my evaluation dataset. I reported the effects of changing the strong links weights to the performance of the system and highlighted the reasoning behind choosing the weights reported in Table 5.2. The results I obtained from evaluating the Wikipedia-extracted features in the application of WSD suggest that the strong links method can give better performance than the term-concepts table.

5.5 Summarization System

In this section, I describe a summarization system employing the features previously extracted from Wikipedia to better enrich the documents representation and enhance the summarization task at hand. The concepts explicitly mentioned within the provided documents are detected and the relationships between them are identified. With the aid of Wikipedia, inferred concepts which are related to the explicitly mentioned documents are also captured and considered along with a list of other features when generating summaries. The network of inferred and explicitly mentioned concepts and the relationships between them provide a mean to quantify their semantic relations with the documents topics and the user's query.

In Figure 5.10 I show the main stages involved during the process. The system accepts as input the documents to be summarized along with the user query. All of the documents to be summarized are first parsed and tokenized with the stop words being removed. Afterwards, the Wikipedia extracted features are used to enrich the representation of the documents. This enrichment process produces several vectors reflecting different aspects of the documents including the important terms, top Exact Match (EM) concepts/related concepts, and the top M related categories. After generating the different vectors, they are passed to the sentences ranking module which in return produces a sorted list of the top candidates to be part of the summary.

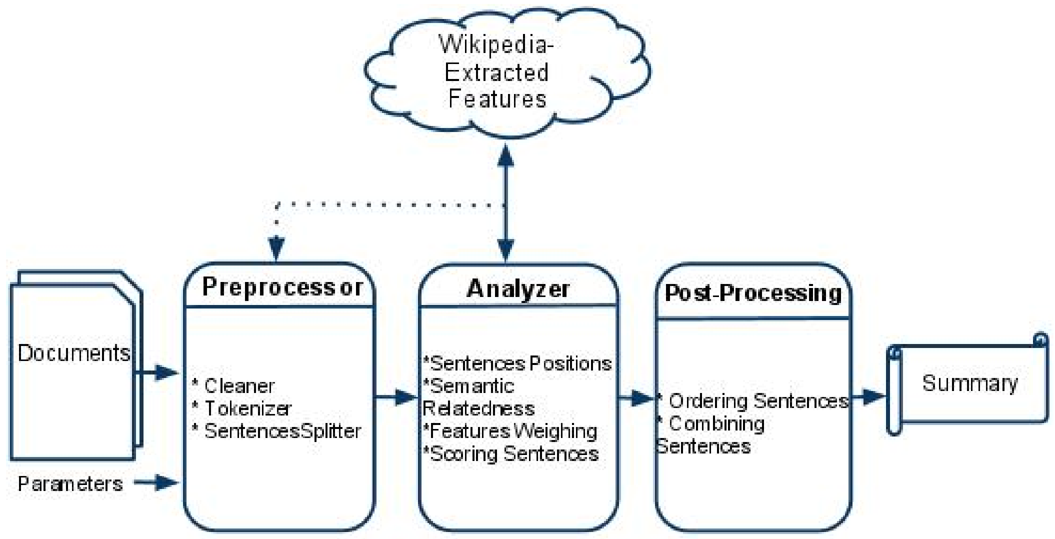


Figure 5.10: **Stages of the Wikipedia-backed Summarizer**

In Figure 5.11, I present the class diagram of the implemented system. The elements *DocumentSet*, *Document*, *Sentence*, *Term*, *Concept* and *Link* are self explanatory. The classes *textProcessor* and *documentPreprocessor* are similar to the ones applied in the WordNet-based summarizer and shown in Figure 4.6 and Figure 4.5. The class *WikiFeaturesRetrieval* inherits the *MySQLBroker* class shown in Figure 5.2 and gives access to the previously constructed features from Wikipedia which are stored in a MySQL database. *ConceptsRelatednessMetric* defines and implements how the relatedness between concepts is computed while *SentencesRlatednessMeasure* computes the relatedness between sentences.

The developed system has been used to participate in the TAC10 summarization task. The evaluation results are reported in a subsequent section. In the next section I give an overview of the main stages involved in the framework followed by the evaluation.

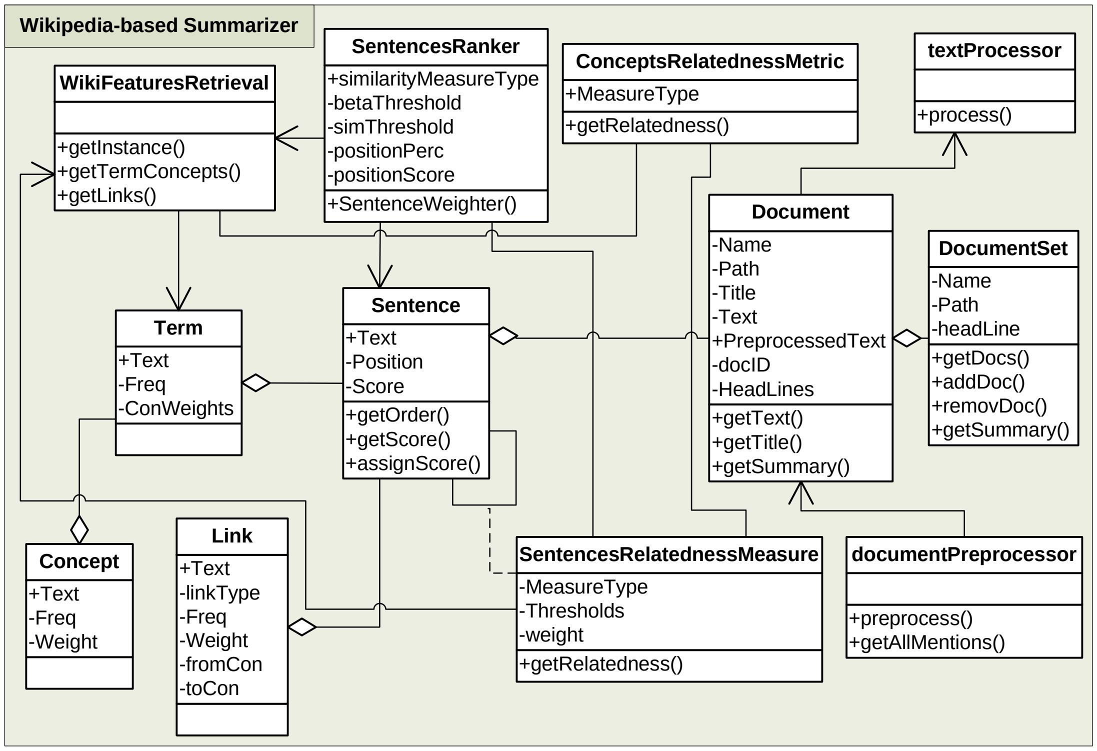


Figure 5.11: Class diagram showing the main classes of the *Wikipedia-assisted Summarizer* package

5.5.1 Preprocessing

The first stage in the framework is to preprocess all fed documents by cleaning and then parsing them to extract the text and topics and then tokenizing the terms and splitting the sentences. Stop words are then removed.

5.5.2 Identifying the Concepts

Two methods have been utilized to detect concepts by employing the built Wikipedia-thesaurus and its extracted features. The first one is through an exact match measure where explicitly mentioned concepts within each sentence are detected. A concept having multiple spellings and synonyms should still be detected by the system as a single concept. This is due to the integration of redirect links within the thesaurus and the mapping algorithm that associates sentences with the concepts they contain. As for ambiguous terms and concepts, the system implements the Weighted Strong Links method that was described in section 5.4.3.3.

In the second method I examine each term within a sentence and replace it with its concepts vector using the term-concepts table. The concepts vector has a weight associated to each concept signifying its relatedness with the term. After forming a concepts vector for each term, I group all concepts vectors within a sentence by summing the scores of the individual concepts which are repeated. This in effect applies word sense disambiguation as relevant concepts are boosted and given a higher score in the merged concepts vector. For example, the concept “Fox” has two meanings: “Fox (Animal)” and “Fox (Broadcasting company)”. Similarly, the concept “Dog” is associated with “Mammals”. In the sentence “A dog attacked a *fox*”, the meaning “Fox (Animal)” is boosted.

5.5.3 Measuring the Relatedness between Concepts

For every explicitly detected concept, it is possible to devise a vector of related articles through the strong links method. The vector would contain the related articles and the weight assigned to each based on the detected link types between them. I compute the relatedness between any two concepts using the cosine measure formula as follows:

$$rel(a,b) = \frac{\sum_{i,j} a_i \times b_j}{\sqrt{\sum_{i=1}^{n} (a_i)^2} \times \sqrt{\sum_{j}^{m} (b_j)^2}} \qquad \textbf{5.17}$$

where a and b are the two concepts to be compared and a_i and b_j are the weights associated with their related articles as extracted from Wikipedia using the Strong Links method.

5.5.4 Measuring the Relatedness between Sentences

Each sentence would have a vector of the concepts detected in it using the exact match method. In addition, it would have another vector of concepts generated from merging its individual terms concepts as extracted from the term-concepts table. When evaluating two sentences, I consider both vectors to compute the relatedness between them. The semantic relatedness is computed by the following formula:

$$Srel(Sent_1, Sent_2) = \frac{rel(A,B)}{PairsCounter} \qquad \textbf{5.18}$$

Where $Sent_1$ and $Sent_2$ refer to Sentence1 and Sentence2 respectively, A is the concepts set in Sentence1, B is the concepts set in Sentence2, and PairsCounter is the number of concepts pairs compared. This formula can be applied to both vectors individually.

5.5.5 Feature Selection

Each sentence is tagged with several features. These features are used to compute a score determining the sentence importance.

Overlap with the Topic/Query: I consider the overlap between each sentence and the topic of its documents set. I take into account the *concepts* overlap when assigning a score to each sentence. Synonyms and concepts with alternative spellings are considered as a single concept in my system with the help of the Wikipedia thesaurus and the custom matcher.

Concepts Dominance: The explicitly mentioned concepts within a document set which are most frequent and the topic concepts are considered to be the most important. When computing a score for each sentence based on this feature, I consider how pertinent the sentence concepts to the important concepts with the document set.

Sentence Position: The system assumes that sentences appearing at the top and bottom of a document have more chances of being important than the rest. Therefore, sentences appearing in the top 20% and the bottom 20% portion of a document are given position scores 50% larger than the others.

5.5.6 Sentences Scoring

As with the WordNet-based summarizer, each sentence is given a score representing its importance. The score for each sentence (i), is simply the linear combination of the weights given for each feature. The formula used for assigning a score to each sentence is:

$$\text{Score(i)} = \frac{(\alpha\, Srel(s_i, \text{T}) + (1-\alpha)\, Srel(s_i, Q))\; n(s_i)\; P(s_i)}{N} \qquad \textbf{5.19}$$

Where:

- N = the total number of sentences
- n(si) = The number of sentences that have semantic relatedness score bigger than a pre-defined threshold value
- P(s) = either 1 for sentences appearing at the top and end of the document, or 0.5 for the rest.
- Srel(s_i ,T) and Srel(s_i ,Q) are for the Semantic Relatedness between the Title and the Query, respectively, and the sentence (i)

The rationale behind the preceding is similar to what was proposed for the WordNet-based summarizer. The main difference here is that the system deals with Wikipedia concepts, and the ones detected within a sentence either directly through EM or indirectly are taken into account when scoring the sentence.

5.5.7 Evaluation

The Text Analysis Conference (TAC) is one of the well-known workshops in the field of Natural Language Processing which provides the infrastructure necessary to evaluate different methodologies with different tasks. In TAC10, I participated in the Guided Summarization task with two different runs. The aim of the task is to provide short summaries for a set of newswire articles. The generated summaries are not to exceed 100 words each. The TAC10 task is different from the ones given in the previous years in that the participants are asked to make a deeper semantic analysis of the source documents instead of simply relying on documents words frequencies to select the important concepts. For this, a list of categories and important aspects for each category are given and it is asked that the summary provided should cover all of the mentioned aspects if possible in addition to any other information related to the topic.

The “update” part of the task is similar to that of TAC08. For a given set of documents, the participants are asked to write two summaries, one for set A and another for set B. A topic statement is provided in addition to the Categories aspects which have been added to the task only this year. The participants are asked to write a summary for set A using the given topic statement and the specified category. For set B, a 100-word update summary is to be generated assuming that the user has already read the set of articles in set A. I participated with two runs. The ids of my runs are 14 and 19. The α parameter was set to 1 for both runs giving higher emphasis to the topic since no user queries are provided. The term-concepts table method was used with run 19 while strong links method was used for 14.

In TAC10, three metrics were used to evaluate systems summaries against refernce summaries prepared by humans: 1) ROUGE which is based on overlapping units, 2) BE which relies on Basic Elements and 3) manual scores for Summaries Content Units (SCU), linguistic quality and responsiveness. In Table 5.5, I show the scores obtained in the automatic evaluations and the ranks of my system in the different evaluations settings. The total number of peers the system is evaluated against is 43. The obtained ranks show an above-average result for all evaluations. The strong links method also gave generally better performance than the one using the term-concepts table.

	Scores			**Ranks**		
Evaluation	**ROUGE2**	**ROUGESU4**	**BE**	**ROUGE2**	**ROUGESU4**	**BE**
Baseline1	0.053515	0.086865	0.029415	30	33	30
Baseline2	0.060965	0.093925	0.035485	25	29	23
My System 14	0.073921	0.112955	0.044357	13	14	10
My System 19	0.073562	0.111683	0.043174	17	15	15
Average	0.061678	0.097744	0.034408			
Best	0.082861	0.120605	0.049935			
Worst	0.002875	0.009935	0.001015			

Table 5.5: **Evaluation results for the Summarization Task showing the scores and ranks of the two submitted runs 14 and 19 relative to other peers**

As for manual evaluations, the system also obtained competitive results, and the ranks are relatively better than with the automatic evaluations. Among the submitted peers by the TAC10 organizers were two baseline summarizers: Baseline1 which is a basic summarizer that returns all leading sentences in the most recent document. Summarizers that rely heavily on choosing lead sentences from test documents have been found in the literature to obtain highly competitive results as reported in [173] and [174], especially when dealing with newswire articles. The second baseline is the MEAD summarizer [103]. It scores sentences according to a linear combination of features including centroid, position and first sentence overlap and considers cross-sentences dependencies. It models all documents as BOW and was also found to obtain competitive results when applied to newswire documents. More details about this summarizer were presented in section 2.4.1. In all evaluations performed, my two submitted runs obtained better performance than both baselines and above average scores. Both baselines rely on BOW-based techniques and perform no training. Comparing the results of my runs, I find that the employing several aspects of Wikipedia resulted in a better performance than employing only the text of Wikipedia articles as was performed with the term-concepts table. The strong-links method benefits from the linking structure within Wikipedia in addition to its categories and the manually crafted links by humans indicating the semantic relatedness between concepts (*See Also* links).

The results of the manual evaluations are shown in Table 5.6. The ranks and scores for the manual evaluation were comparatively better than those of the automatic evaluation. This was expected as the summarizer is extractive and no modifications were made to the extracted sentences on the submitted runs. Overall, the ranks for the different evaluation measures appear to be competitive especially when taking into consideration the fact that the system is unsupervised and does not require any training. The obtained results indicate that summarizing with the strong links method give better performance than when using the term-concepts table.

	Scores		**Ranks**	
Evaluation	**Linguistic Quality**	**Overall Responsiveness**	**Linguistic Quality**	**Overall Responsiveness**
Baseline1	3.6955	2.098	1	30
Baseline2	2.7065	2.489	30	23
My System 14	3.1735	2.6635	10	13
My System 19	3.1195	2.6305	11	17
Average	2.800802	2.33336		
Best	3.6955	2.9455		
Worst	1.163	1.0975		

Table 5.6: Manual Evaluation Results for the submitted runs in TAC10

To better understand the reasons why the strong links approach worked better for summarization than the other method relying on term-concepts, I examined the generated summary for each method on document set D1001A and analyzed the top generated concepts for the top sentences in each. The generated summaries for each method along with the top concepts detected by each are illustrated in Table 5.7. Comparing the detected concepts for each method, I find that some of the detected concepts with the term-concepts table method do not seem to be much related to the articles being summarized. For example, the concept *Swat, Pakistan* was found to be among the top contributing concepts when generating the summary. This was in part due to the chosen context size for the term *SWAT* being larger than it should be. The article *Swat, Pakistan* in Wikipedia included terms such as *killing*, *kill*, *death*, *school* and *student* which appear to be among the surrounding terms for the *SWAT* term in the provided test documents. This noise caused by the large context size has affected the performance negatively for the term-concepts table method. On the other hand, the strong links method relied on the explicitly mentioned concepts within the test documents and inferred the top related concepts to detect the theme of the test documents and form the summary at the end. This had the effect of correctly identifying the right concept of the term *SWAT*.

Term-Concepts table	**Summary:** `The day that Columbine High School students are to return to class has been delayed because so many have been attending funerals for students killed in the April 20 massacre, an administrator said Tuesday. Two days earlier, a massacre by two students at Columbine High, whose teams are called the Rebels, left 15 people dead and dozens wounded. Harris and Klebold, who authorities say planned the massacre for more than a year, have been portrayed by classmates as outcasts from the popular students at Columbine.` **Top Concepts:** Eric Harris and Dylan Klebold, Administration of Business, High School, Student, School , Swat, Pakistan, Left-wing Politics, Funeral, Columbine, Colorado, Death, Day School, The Massacre, Federal Government of the United States, Popularity
Strong Links	**Summary:** `The sheriff's initial estimate of as many as 25 dead in the Columbine High massacre was off the mark apparently because the six SWAT teams that swept the building counted some victims more than once. The day that Columbine High School students are to return to class has been delayed because so many have been attending funerals for students killed in the April 20 massacre, an administrator said Tuesday. Two days earlier, a massacre by two students at Columbine High, whose teams are called the Rebels, left 15 people dead and dozens wounded.` **Top Concepts:** Eric Harris and Dylan Klebold, High School, School, Student, Columbine High School, Funeral, Columbine High School Massacre, Murder, Local Government, Funeral, SWAT, Killed in Action, Rebellion, Homicide, Columbine, Colorado, Authority

Table 5.7: Sample summary output for the two submitted runs on the document set D1001A in TAC10

In order to get a deeper view at the performance of the system, I tried to obtain the ROUGE scores my best run obtained for each topic in the evaluated documents sets A and B. I then found the rank of my system in each topic, and adjusted it with 1/(1+log(r)) where *r* is the rank of my system. I applied that formula to my ranks in each topic and obtained the results illustrated in Figure 5.12, Figure 5.13, Figure 5.14, and Figure 5.15 where 1 means the best rank was obtained while 0 means the worst.

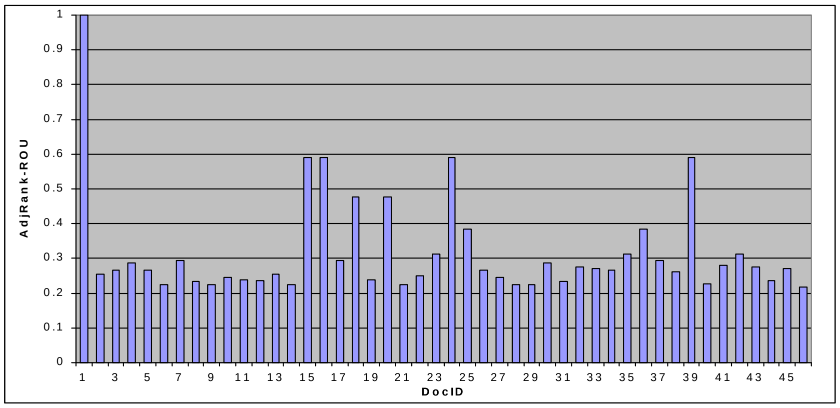


Figure 5.12: Adjusted Ranks Obtained as applied on all Topics in Set A with ROUGE2

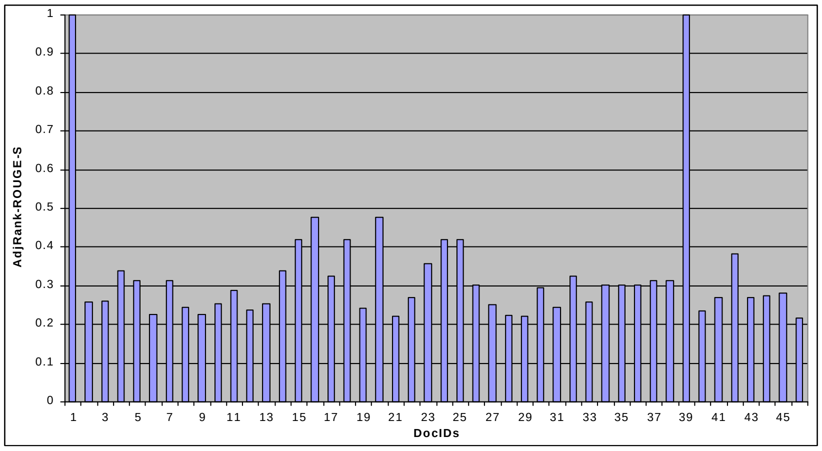


Figure 5.13: Adjusted Ranks obtained for Topics in Set A with ROUGE-SU4

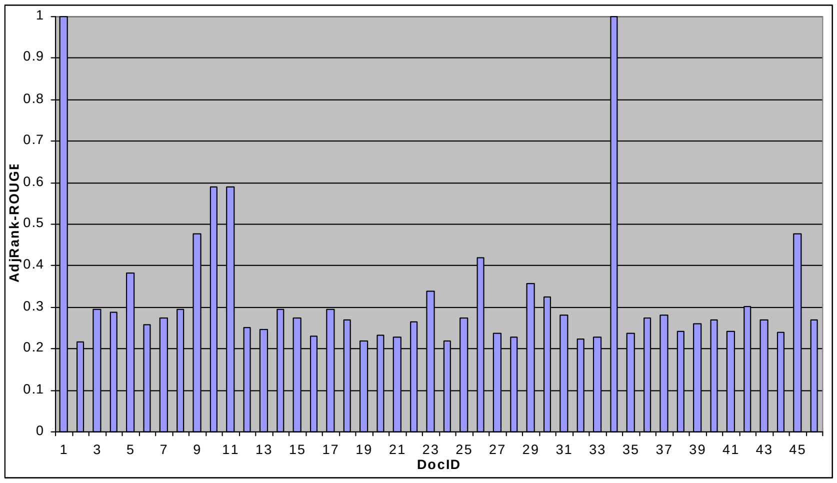


Figure 5.14: Adjusted Ranks for Topics in Set B with ROUGE-2

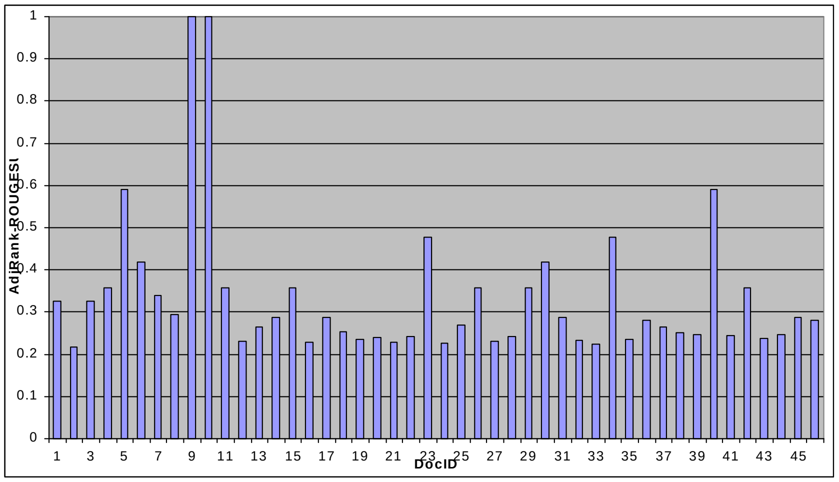


Figure 5.15: Adjusted Ranks for Topics in Set B with ROUGE-SU4

The results shown in the figures demonstrate that my system obtained the best ranks for topics 1 and 39 from Set A followed by topics 15, 16 and 24. The worst performing topics were 14, 28, 29 and 46. In Set B, the best ranks obtained were for topics 1, 9, 10 and 34 while the worst were for 2, 19 and 24.

When examining the summaries of the low ranking topics, it was observed that in many cases the sentences added to the summaries by my runs were long and comprised nearly half of the summary length. Introducing means to simplify or compress these long sentences without removing the important information they contain could allow for adding even more content to the summary without going beyond the summary length limit. In addition, some summaries contained sentences having redundant information. Handling redundancy is likely to introduce improvements to the summarizer. In the next chapter, I investigate handling these limitations through the introduction of an SSM module and present its effects on the system.

5.6 Conclusion

In this chapter, I explained how Wikipedia can benefit a summarization system through the abundant and diverse knowledge it contains. I described the need for a module to extract important content from Wikipedia and transform them into a useful format that can be used in NLP-based applications. I introduced my Wikipedia Features Extractor and detailed its implementation. Afterwards, I examined the usage of the extracted features in the application of WSD and evaluated the performance of the system. Following that, I described how the extracted features can help enrich the representation of documents and aid in the task of Automatic Summarization. In addition, I introduced measures for computing the relatedness between any concepts, text fragments or sentences. The results of the evaluations performed on the summarization system gave competitive results and shed light on potential areas of improvement such as simplifying sentences and handling redundancy. In the next chapter I describe my attempt in that direction and detail my findings.

Chapter 6

Sentences Simplification for Automatic Summarization

In this chapter, I emphasize the need for conserving space within sentences by introducing a Sentences Simplification Module (SSM). The module is aimed at shortening the length of sentences via either splitting or compression. I describe how the module is integrated with the Wikipedia-based summarization framework. I highlight the performance differences obtained from introducing such a module by running a series of evaluations.

6.1 Overview

The goal of summarization systems is to provide summaries containing as much information as possible meeting the user's needs within a confined space determined by the previously-set summary limit. Since the developed summarization systems that were described in the previous chapters are extractive, they are inherently limited to only the sentences that exist in the original test documents. A sentence may contain important information in part and non-relevant information in another. Also, a sentence that may be central to the theme of a document may contain a mixture of new and redundant data to what is already available in the summary. It is therefore necessary to enforce a methodology that allows for conserving more space within sentences and including as much content as possible without sacrificing novelty or imposing redundancy. For this, it is probably best to view how linguists and discourse analysts perform their studies on text and sentences in particular. Their analysis usually begins by separating clauses and phrases within a sentence to identify their features and properties [175]. In this chapter, I aim to apply a similar step by introducing automatic sentences simplification to the

Wikipedia-based summarization framework for the purpose of better information extraction.

Sentence Simplification can be defined as the operations used to simplify a sentence structure and grammar and transform it into a simpler form that preserves the underlying meaning or information. The changes being made to the original sentence include reordering or dropping terms/phrases within the sentence, splitting the sentence, merging its clauses or replacing terms/phrases with others which are either shorter or simpler. The focus here is to implement sentences simplification by splitting and compressing sentences.

Sentences Simplification has been applied in the literature to different applications with different motivations. In [176] and [177], it has been used to create sentences that are easier to read for humans and new language learners in particular. Their target was to create sentences which are grammatically correct, short and cohesive. Sentence simplification has also been applied in summarization systems to shorten the length of sentences. The CLASSY summarization system described in [33], [178] and [179] for example employs a sentence compression module by applying a set of rules to all sentences before choosing the summary candidate sentences. Cut-and-Paste in [45] and other summarizers in [180], [181] and [182] employ a similar approach. In [22], the authors applied a sentences compression module after choosing the candidate sentences in a post-processing step.

In the mentioned systems, compression is applied independently as a separate process to all sentences and is not affected by how candidates are selected. It is possible that two sentences containing an important piece of information are compressed by removing that important content from both. It would be optimal to remove the redundant info from only one sentence while keeping it in the other if both sentences were to be included in a summary. Implementing this would require a system having the capability of applying a dynamic set of rules to different sentences based on what is already contained in the summary.

It was suggested in a pilot study in [183] that summarization systems implementing compressions would have an edge over those that do not if the compression took into account the different references and relationships among sentences. In [184], an approach was applied to tackle the mentioned issue. Their system would apply a set of rules for trimming sentences and creating multiple versions of each sentence using the rules described in [24]. After creating multiple compressed versions of each sentence, the core summarizer would consider all versions of each sentence as potential summary candidates. To choose the optimal candidate, the system would check the redundancy of the sentence against the current summary and select the least redundant. Another system with a similar approach was implemented and described in [185]. Both systems emphasized preserving only important content and do not necessarily preserve semantic content. Redundancy checking is based on BOW methods and do not capture the semantics similarity and relatedness between the different sentences and the concepts they carry. In my system, I also employ a similar methodology by producing multiple simplified versions of sentences. However, the focus here is rather on simplifying and compressing sentences while in the same time preserving as much relevant semantic content as possible. For that, the Sentences Simplification Module (SSM) I developed is integrated with the Wikipedia-based summarizer described in the previous chapter and an iterative process is added.

6.2 Sentences Simplification Module

Before simplifying a sentence, it is necessary first to devise some means for interpreting its text. The interpretation can be syntactic, lexical or semantic. Focus here is on syntactical interpretation of sentences. For that, dependency tree of the sentence is drawn with help from Stanford's Parser[14] which adopts the Penn Treebank conventions[15]. With the tree drawn, one can apply any set of rules to make the changes desired to the sentence. In effect, the rules may allow for simplifying a long and complicated sentence through compression or splitting into several sentences. The simplified sentences, along with the original, become summary candidates and the choice is based on the contained semantic information in each and the redundancy and relevancy to what is already in the current state of the summary. Factors like sentences length (before and after the split/compression) and the existence of dominant concepts/words/phrases help play a role in making decisions dynamically. Figure 6.1 gives an overview on the simplification process. It can be noted that the parser produces two types of output: the phrase structure tree and the dependencies which are otherwise known as grammatical relations. The sentence simplification module outputs the simplified versions of the sentence which are used in combination with the original sentence as inputs in other parts of the system to choose the best summary candidate from the list.

[14] http://nlp.stanford.edu/software/lex-parser.shtml

[15] Phrase structure annotation conventions used when the parser was trained. More details are available in the appendix

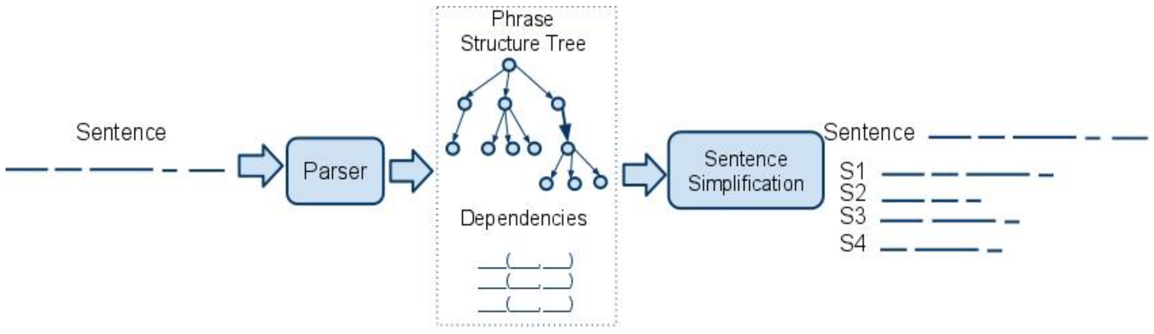


Figure 6.1: Overview of Sentences Simplification Module

In complex sentences, facts, events and statements can be presented with various linguistic constructions. I present an example illustrating this by the sentence:

The latest earthquake, the sixth this year, passed Nanchang in east of China , and waters were rising in Putian, in east China's Fujian province , on the middle reaches of the Fuzhou, state television reported last Sunday.

In Figure 6.2, the types of the different clauses and phrases contained within that sentence are labelled. After applying the simplification process to that sentence, the results I obtain are the sentences shown in Figure 6.3. In the next two subsections I describe the two main processes involved in SSM, namely Sentences Splitting and Sentences Compression.

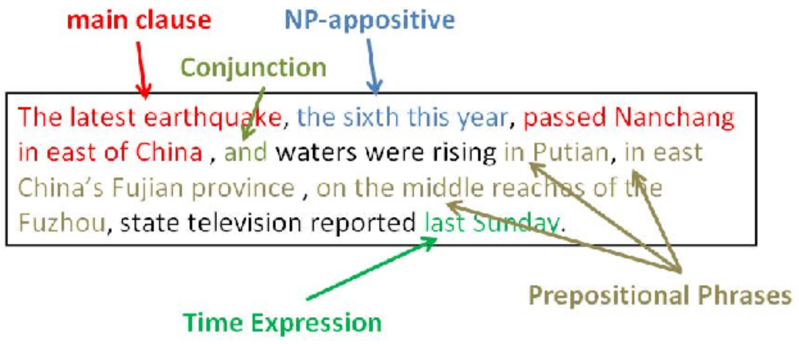

main clause
NP-appositive
Conjunction
The latest earthquake, the sixth this year, passed Nanchang in east of China , and waters were rising in Putian, in east China's Fujian province , on the middle reaches of the Fuzhou, state television reported last Sunday.
Prepositional Phrases
Time Expression

Figure 6.2: An example of a complicated sentence with labelling of different clauses types

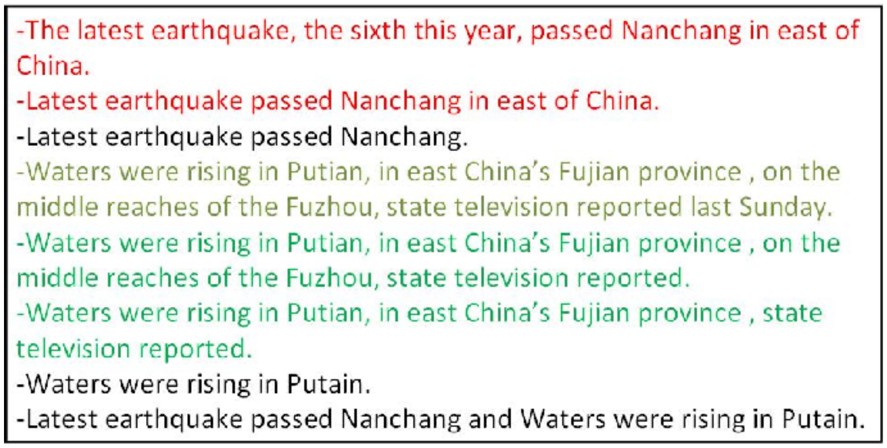

-The latest earthquake, the sixth this year, passed Nanchang in east of China.
-Latest earthquake passed Nanchang in east of China.
-Latest earthquake passed Nanchang.
-Waters were rising in Putian, in east China's Fujian province , on the middle reaches of the Fuzhou, state television reported last Sunday.
-Waters were rising in Putian, in east China's Fujian province , on the middle reaches of the Fuzhou, state television reported.
-Waters were rising in Putian, in east China's Fujian province , state television reported.
-Waters were rising in Putain.
-Latest earthquake passed Nanchang and Waters were rising in Putain.

Figure 6.3: Result of the simplification process when applied to a complicated sentence

6.2.1 Sentences Splitting

Sentences with different syntactic formats would need to be handled differently with different rules applied to achieve the splits desired. Based on the generated parse tree and grammatical relations for the original sentence and the boundary terms found in a sentence, it is decided what rule to apply. The boundary terms are chosen to be *who*, *which*, *that*, *and* and *or*. Many of the rules mentioned here have been applied in the literature in different studies for different systems including [176], [186] and [187]. The following are few scenarios illustrating the patterns being detected and processed by SSM:

<u>Example 1</u>: *The man who ate the poisoned food died yesterday.*

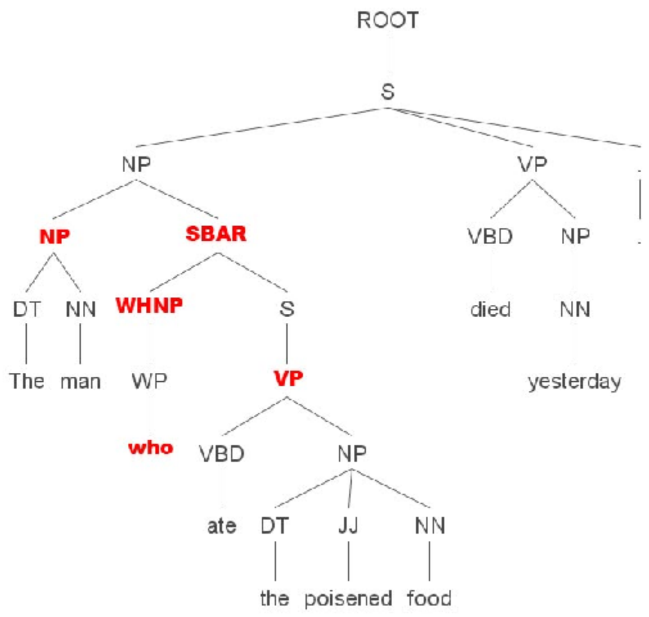


Figure 6.4: Parse Tree of the first Example for Splitting Sentences

Used Modifiers:
rcmod(man-2, ate-4)
Generated sentences:
The man died yesterday. The man ate the poisoned food.

In the above example, the boundary term *who* appears before the phrase *ate the food yesterday*. When viewing the phrase structure tree for the sentence, a phrase of type *SBAR* which has *WHNP* as one of its children (or grandchildren) is detected. This indicates that a potential split is in place. The split is achieved by separating the *SBAR* clause from the rest of the sentence resulting in two sentences, with one being incomplete. These two sentences are:

The man died yesterday.
who ate the poisoned food.

To complete the second sentence, I look at the phrase proceeding the boundary term *who*, which is of type *VP*. With the relation *rcmod(man-2, ate-4)*, one can tell that *man* is the word that should be preceding the main verb in the incomplete sentence.

<u>Example 2</u>: *I have read the books which you bought last week.*

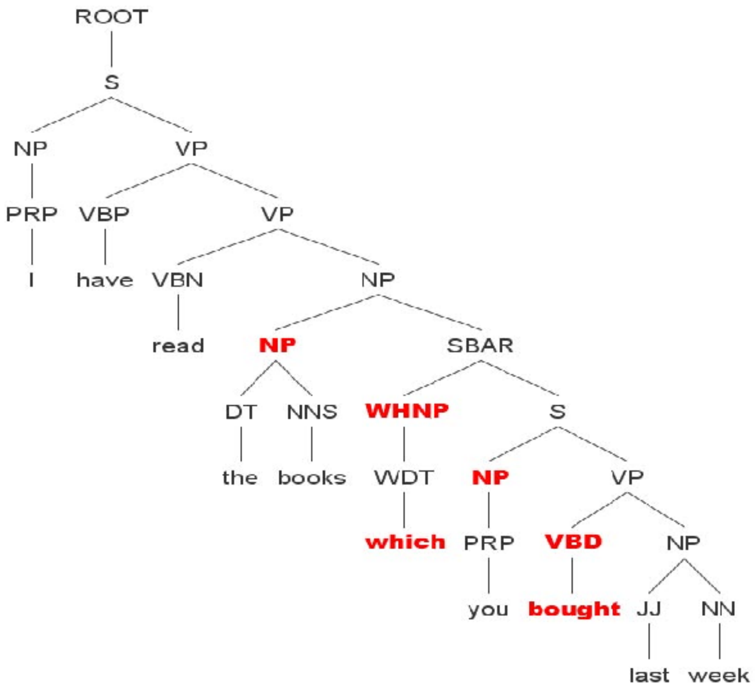


Figure 6.5: Parse Tree of the Second Example for Sentences Splitting

Used Modifiers:
rcmod(books-5, bought-8)
Generated Sentences:
I have read the books. You bought the books last week.

In this sentence, the boundary term *which* is proceeded by the Noun Phrase (*NP*) *you*. For this scenario, I find the following *VBD*, which is *bought*, and use the relation *rcmod* to find the subject that is being referred to by that verb. I obtain *books* as the answer. The *NP* of *books* is used to complete the second sentence.

<u>Example 3:</u> *I have read the books that you bought last week.*

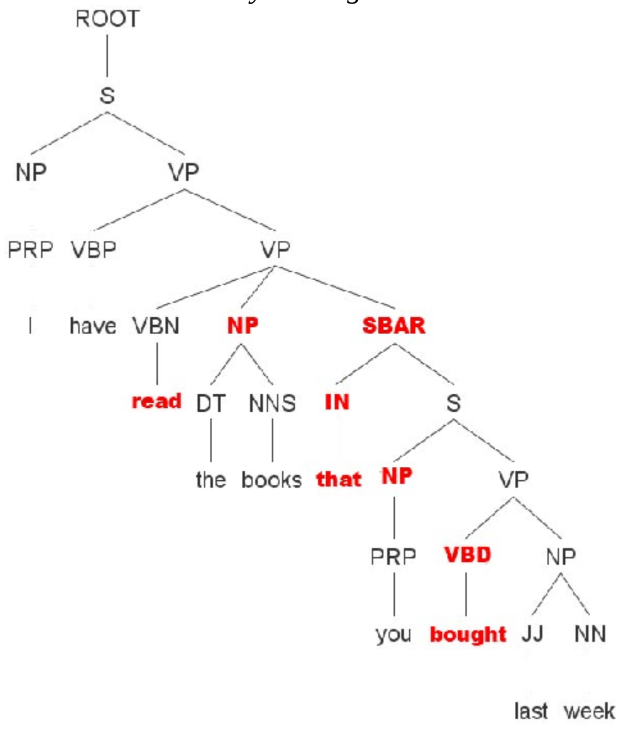


Figure 6.6: Parse Tree of the Third Example for Sentences Splitting

Used Modifiers:
dobj(read-3, books-5)
ccomp(read-3, bought-8)
Generated Sentences:
I have read the books. You bought the books last week.

The boundary term in this example is *that*. It is found underneath an *SBAR* and has an *NP* following it. The *VBD bought* is used with the relation *ccomp* to find its complement: *read*. The *dobj* relation is then employed to specify the direct object of the verb which is *books* in my example.

Example 4: *I have read the books you bought last week.*

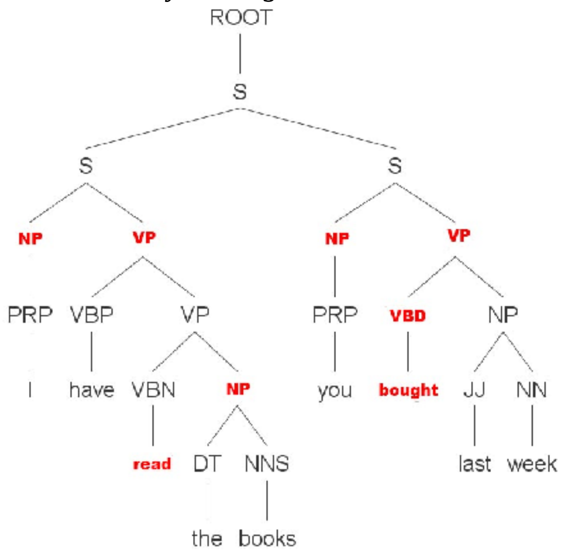


Figure 6.7: Parse Tree of the Fourth Example for Sentences Splitting

Used Modifiers:
dep(read-3, bought-7)
dobj(read-3, books-5)

Generated Sentences:

I have read the books. You bought the books last week.

This example does not contain a boundary term. Two *S* clauses appear with *NP* and *VP* for each. The segmentation takes place by separating the clauses from each other. Since the verb *bought* is dependent on *read*, I use the relation *dobj* to determine the *NP* following *bought*.

Example 5: *The team won the golden medal and achieved the highest team score of the season*

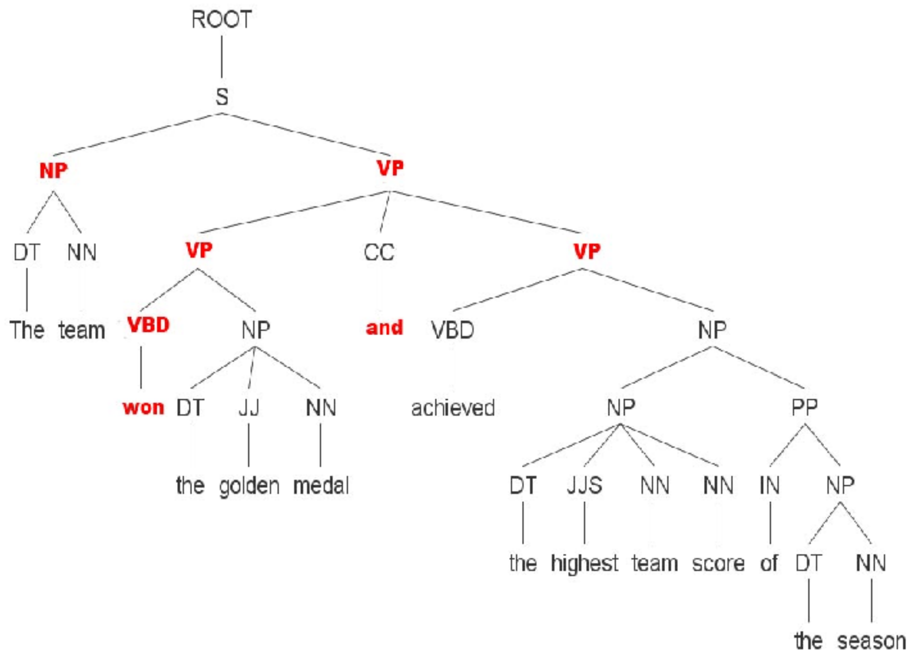


Figure 6.8: Parse Tree of the Fifth Example for Sentences Splitting

Used Modifiers:

conj_and(won-3, achieved-8)
nsubj(won-3, team-2)

Generated Sentences:

The team won the golden medal. The team achieved the highest team score of the season.

The boundary term in this example is *and* which separates two *VPs*. The relation *conj_and* is used to link the main verb of the second clause with the first and thus find the shared object.

<u>Example 6</u>: *The infected rabid fox eventually dies, but a simple scratch can spread the virus to other animals or people*

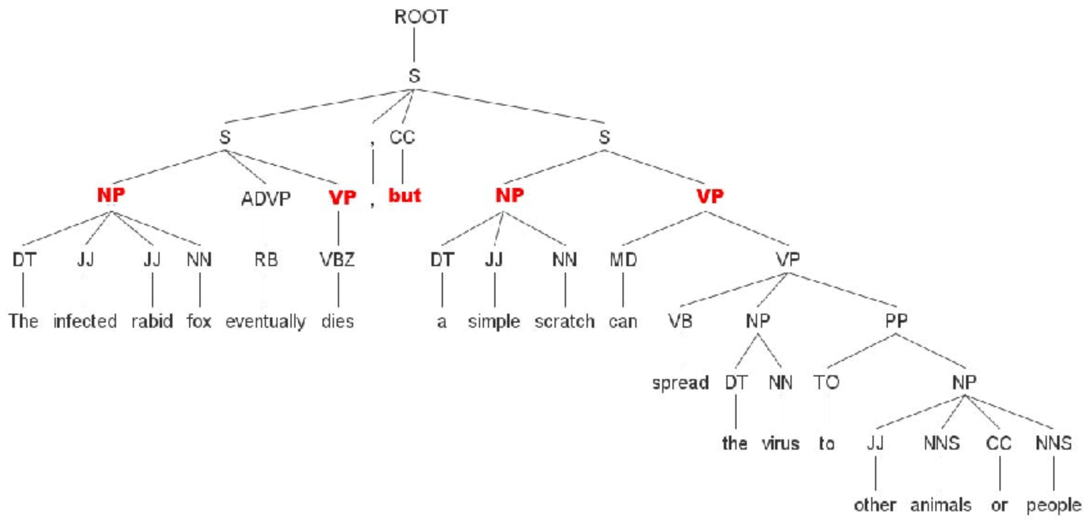


Figure 6.9: Parse Tree of the Sixth Example for Sentences Splitting

Used Modifiers:
conj_but(dies-6, spread-13)

Generated Sentences:

The infected rabid fox eventually dies. A simple scratch can spread the virus to other animals or people.

The boundary term in this example is *but*. It's located between two clauses of type *S*. Both clauses have *NP*s followed by *VP*s as their children. This pattern triggers a possible split which is achieved with help from the relation *conj_but* to construct the new sentences.

6.2.2 Sentences Compression

I adopt the Trimmer algorithm described in [24] and [92] by applying some of its syntactic compression rules. The original Trimmer algorithm aims to transform sentences into Headline-style phrases by detecting patterns in a sentence parse tree and removing certain nodes from the tree based on the applied rule and detected pattern. The goal here is to generate sentences, not headlines, which are short, cohesive and grammatically correct. Therefore, I implement only some of the rules that would be most suitable for producing complete sentences and discard those resulting in headlines-styled text. At first, the rules are applied to a sentence independently. And then in a second iteration, they are all applied to the sentence in order with the output of one rule being fed to the next. The aim of both operations (applying rules independently and in order) is to produce as many compressed and valid versions of a sentence as possible. The rules being used are the following:

1- <u>Keep Leftmost S Root and Remove the rest</u>

Keep the leftmost S Root which has both NP and VP in the sentence and the remove the rest.

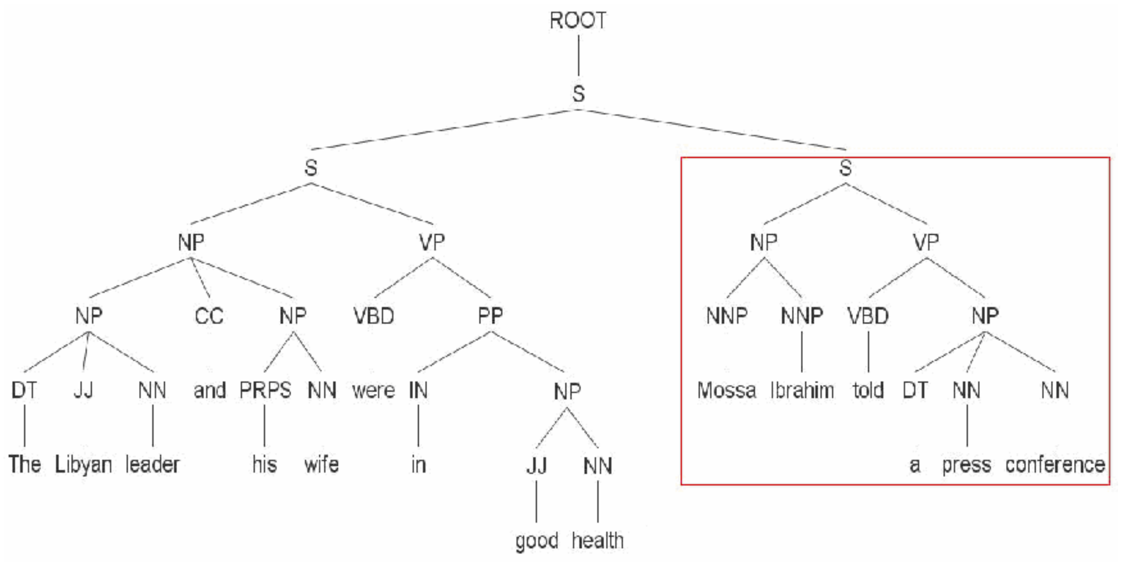


Figure 6.10: Keeping the leftmost S Root and removing its siblings

<u>Original Sentence:</u> The Libyan leader and his wife were in good health, *Mossa Ibrahim told a press conference*.

<u>After compression:</u> The Libyan leader and his wife were in good health.

2- <u>Remove Time Expressions</u>

Remove temporal expressions from sentences. This is achieved by deleting the *PP* node which has an *NP* child with a Time word as one of its leaves. The deletion takes place by removing the Prepositional Phrase (*PP*) with all of its children.

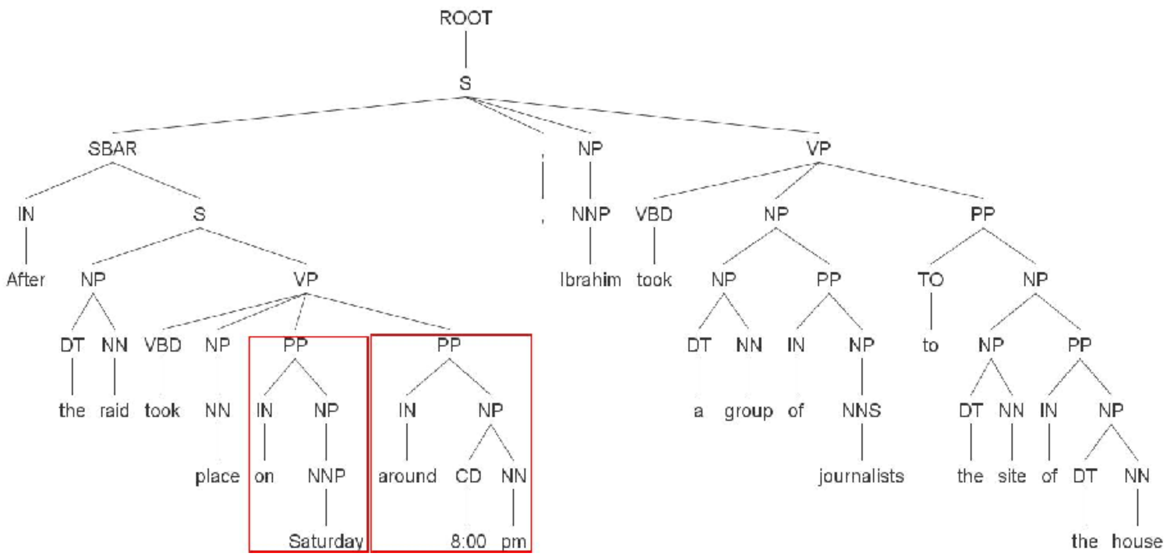

ROOT
S
SBAR , NP VP
IN S , NNP VBD NP PP
After NP VP Ibrahim took NP PP TO NP
DT NN VBD NP PP PP DT NN IN NP to NP PP
the raid took NN IN NP IN NP a group of NNS DT NN IN NP
place on NNP around CD NN journalists the site of DT NN
Saturday 8:00 pm the house

Figure 6.11: Removing Time Expressions for Sentences Compression

<u>Original Sentence</u>: After the raid took place *on Saturday around 8:00 pm*, Ibrahim took a group of journalists to the site of the house.

<u>After Compression</u>: After the raid took place, Ibrahim took a group of journalists to the site of the house.

3- <u>Remove Conjunctions</u>

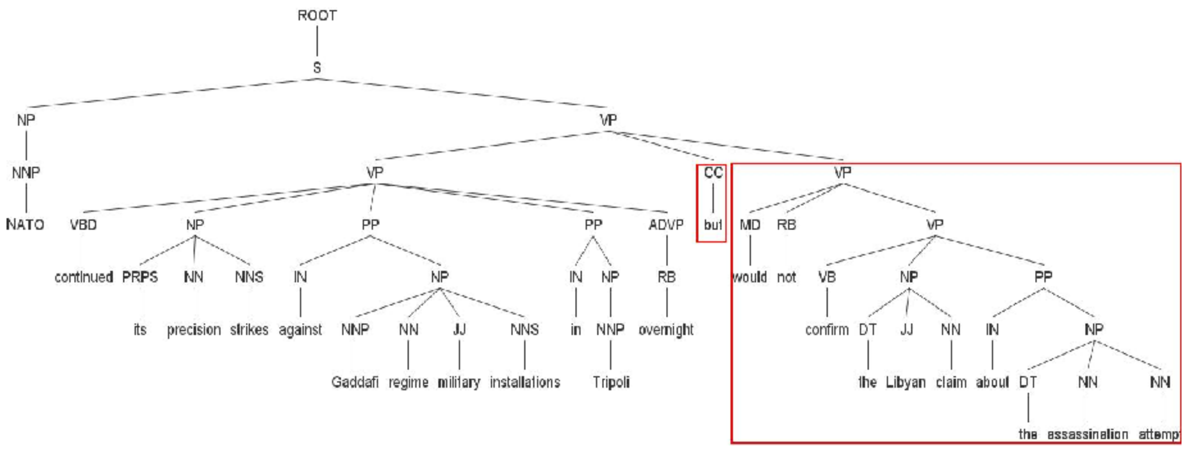

ROOT
S
NP VP
NNP VP CC VP
NATO VBD NP PP PP ADVP but MD RB VP
continued PRPS NN NNS IN NP IN NP RB would not VB NP PP
its precision strikes against NNP NN JJ NNS in NNP overnight confirm DT JJ NN IN NP
Gaddafi regime military installations Tripoli the Libyan claim about DT NN NN
the assassination attemp

Figure 6.12: Removing Conjunctions for Sentences Compression

For any sentence containing *AND* or *BUT* as a *CC*, I remove the preceding phrase of *BUT* and proceeding for *AND.*

<u>Before Compression:</u> NATO continued its precision strikes against Gaddafi regime military installations in Tripoli overnight *but would not confirm the Libyan claim about the assassination attempt.*

After Compression: NATO continued its precision strikes against Gaddafi regime military installations in Tripoli overnight.

4- Remove Complements

I remove *IN* nodes which have the term *that* as their leaves.

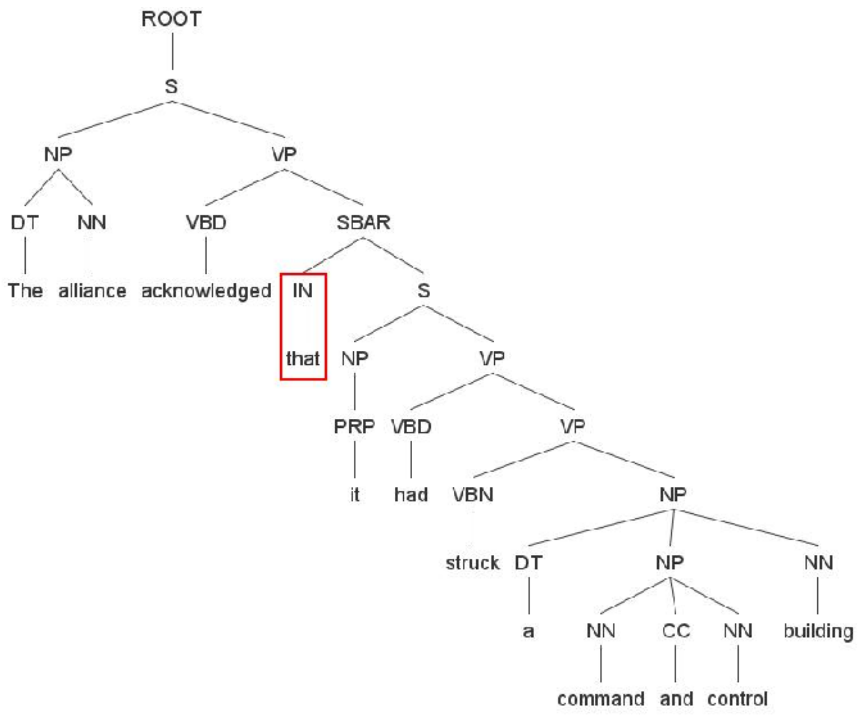


Figure 6.13: Removing Complements for Sentences Compression

Before Compression: The alliance acknowledged *that* it had struck a command and control building

After Compression: The alliance acknowledged it had struck a command and control building

5- <u>XP over XP</u>

XP here refers to either *NP* or *VP*. For first *NP*-over-*NP* or *VP*-over-*VP* where inner *XP* is the first and leftmost child, I keep the left child and remove all of the child siblings. Note that the child *XP* must be the first and leftmost child of the parent *XP* for the rule to apply.

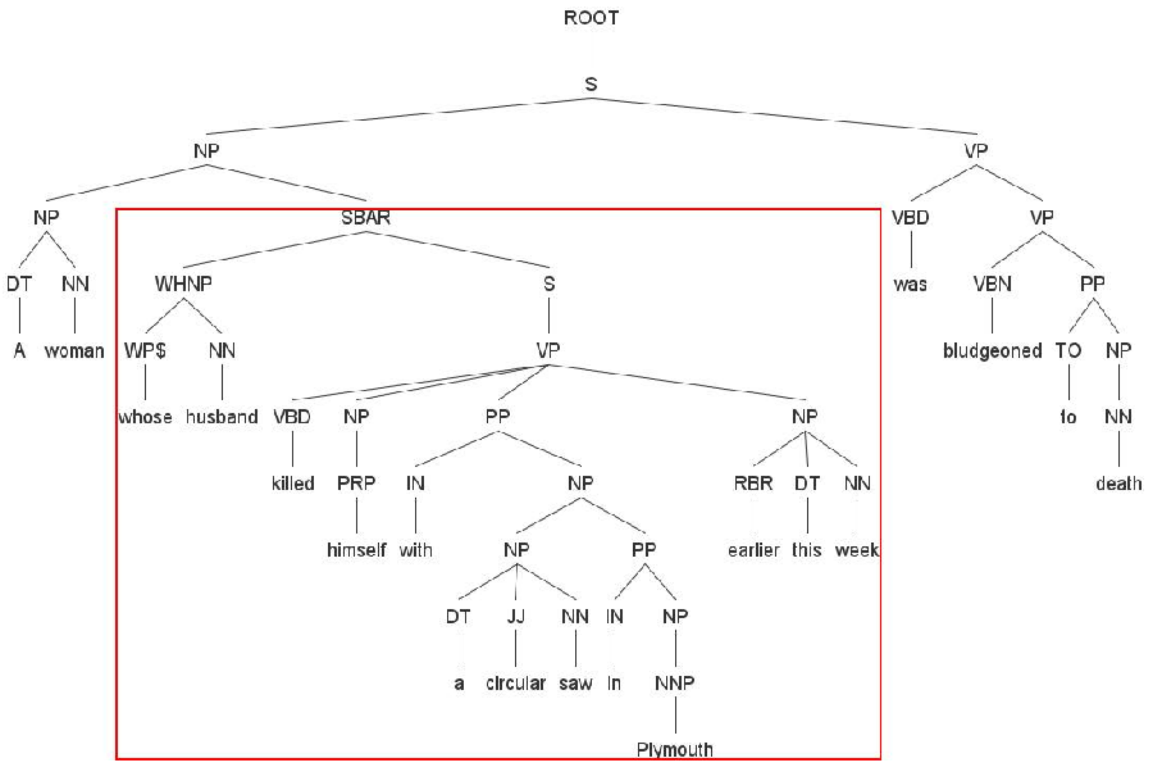


Figure 6.14: Removing Complements for Sentences Compression

<u>Before Compression:</u> A woman *whose husband killed himself with a circular saw in Plymouth earlier this week* was bludgeoned to death.
<u>After Compression:</u> A woman was bludgeoned to death

6- <u>Remove PP under SBAR</u>

PP expressions appearing under *SBAR*s are removed.

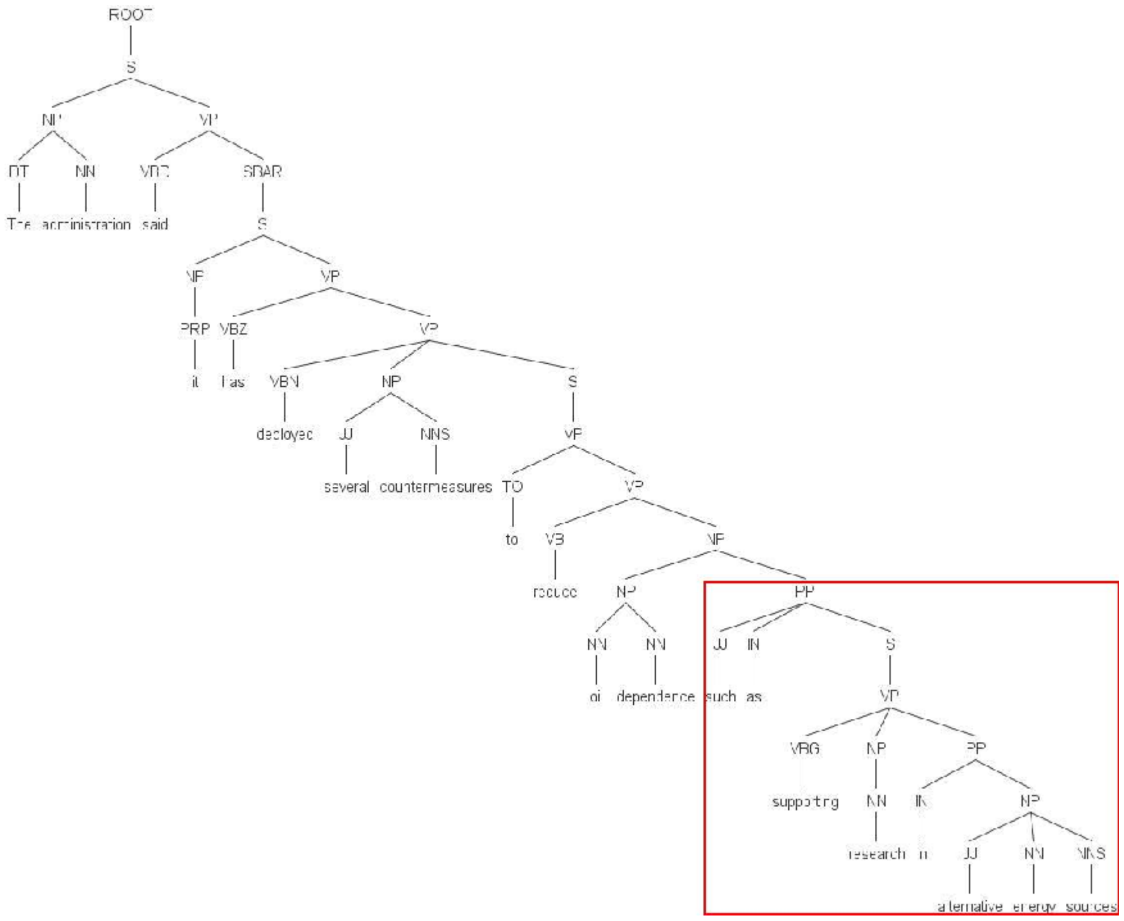


Figure 6.15: Removing *PP*s under *SBAR*s for Sentences Compression

<u>Before Compression:</u> The administration said it has deployed several countermeasures to reduce oil dependence *such as supporting research in alternative energy sources*.
<u>After Compression:</u> The administration said it has deployed several countermeasures to reduce oil dependence.

7- Remove SBAR

I remove *SBAR*s in this step as illustrated in the following example.

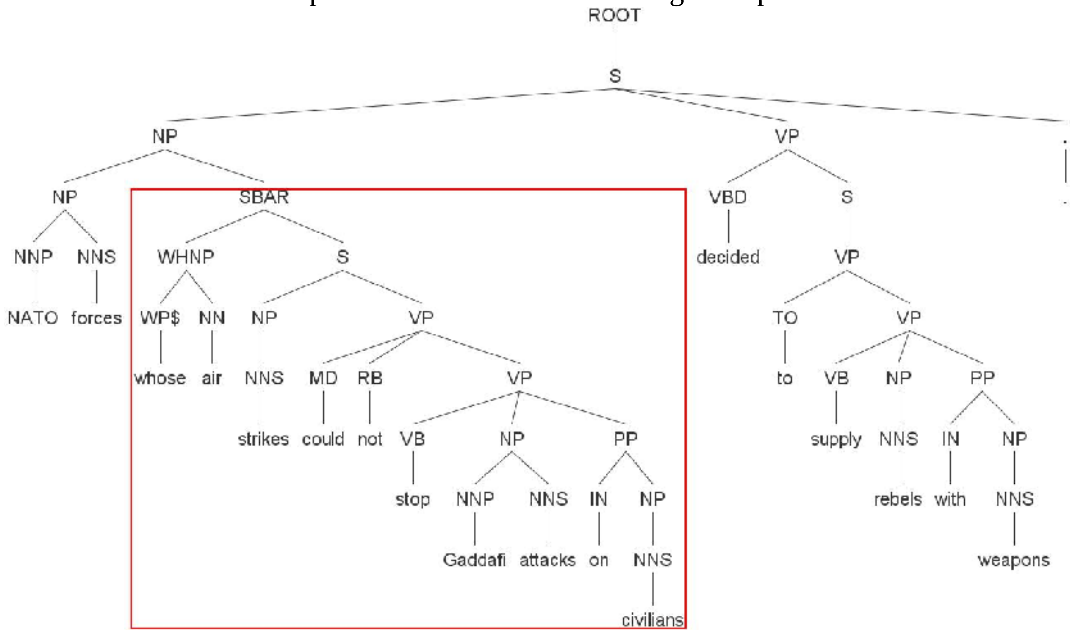


Figure 6.16: Removing *SBAR*s for Sentences Compression

<u>Before Compression:</u> NATO forces *whose air strikes could not stop Gaddafi attacks on civilians* decided to supply rebels with weapons.

<u>After Compression:</u> NATO forces decided to supply rebels with weapons.

8- <u>Remove PP</u>

I remove *PP* nodes in this rule.

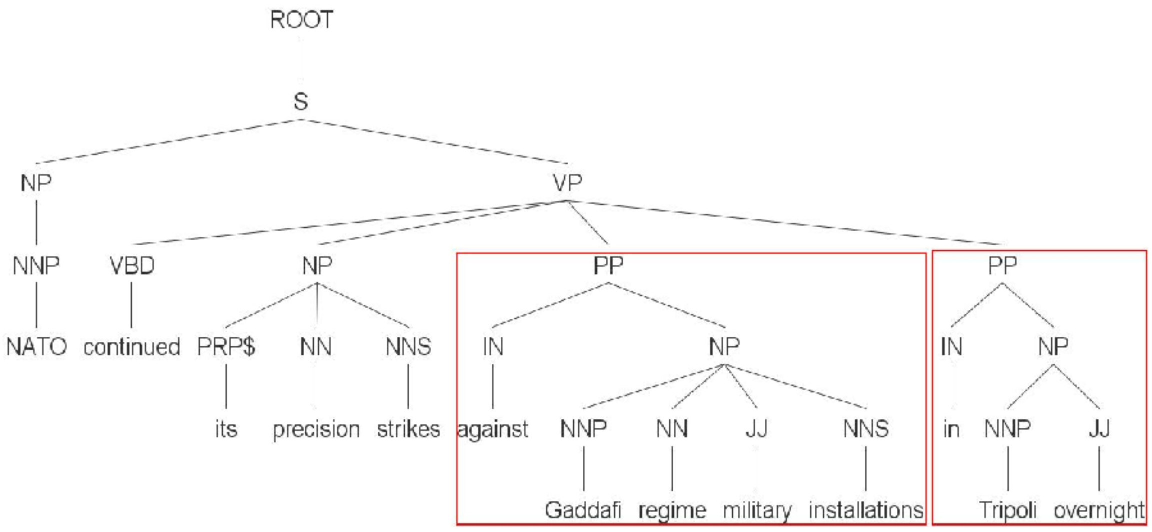


Figure 6.17: Removing *PP*s for Sentences Compression

<u>Before Compression:</u> NATO continued its precision strikes *against Gaddafi regime military installations in Tripoli overnight*.

<u>After Compression:</u> NATO continued its precision strikes.

6.3 Summarization Methodology

The basic summarization methodology here relies on a modified version of the Wikipedia-assisted summarizer that utilizes the strong weighted links approach described in the previous chapter. The major change is in the introduction of SSM and an iterative process handling redundancy. Figure 6.18 shows the architecture of the updated summarizer with the iterative process inside the box. After preprocessing the documents, clusters of sentences are formed with each cluster having the original sentence in addition to its simplified versions as generated from SSM. Afterwards, all sentences are given a score using the weighted links method. Then, an iterative process is applied after which a summary is produced.

The approach I use here assumes that an optimal summary would contain the largest amount of the most useful concepts within a limited space. This is implemented in the system through the introduction of an iterative process enforcing this idea. In the previous approach described in section 5.6, each sentence was given a score signifying its importance based on a set of features: overlap with topic/query, concepts dominance and sentence position. With the iterative process employed here, two of the mentioned features become dynamic, namely overlap with topic/query and concepts dominance. After each iteration, the top ranking sentence is added to the summary and its concepts are identified. The identified concepts are then removed from the source documents and all remaining sentences are rescored. The iterative process can be summarized by the following steps:

1- After scoring all sentences for the first time, I obtain a ranked list of candidate sentences with the top being with the highest score.
2- I remove the top highest scoring sentence from the *Candidate Sentences List (CSL)* and add it to the summary. Only one sentence should exist in the summary at this stage.
3- The cluster of the sentence that was just included in the summary is added to a *Sentences Exclusion List (SEL)*. The cluster should contain the non-simplified version of the sentence in addition to all of its simplified versions.
4- I detect all the *concepts* present in the sentence that was just added to the summary and add them to a *Concepts Exclusion List (CEL)*.
5- I re-score all remaining sentences taking two factors into account: First, sentences in *SEL* should be ignored. Second, any occurrence of a concept that exists in *CEL* should be ignored too.
6- Add the highest scoring sentence to the summary and verify the summary length does not exceed the given limit. If it does not, go to step 3. Otherwise go to the post-processing stage and produce the summary.

Note that in step 5, redundancy is implicitly enforced by counting concepts only once and preferring sentences with a high density of concepts. Simplified sentences that are short and contain important and relevant concepts would still be selected as the approach ensures that no concept repetition within the summary takes place.

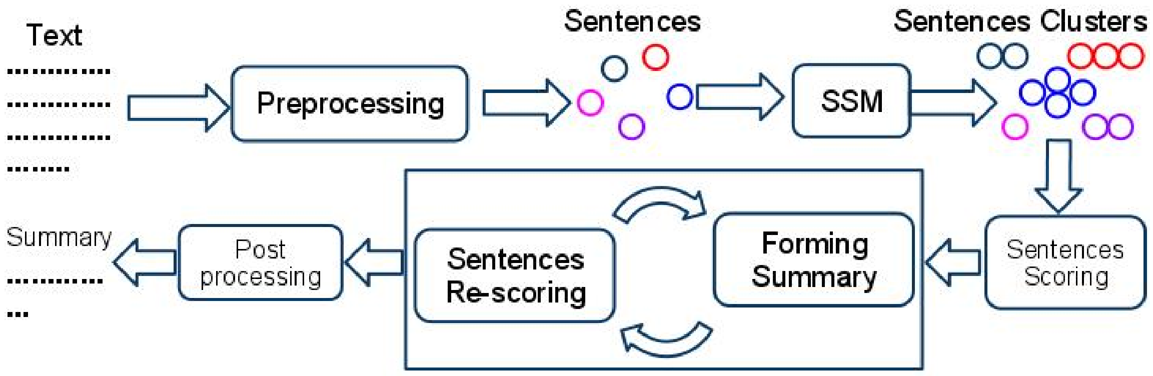


Figure 6.18: Architecture of the SSM-based Summarizer

6.4 Evaluation

To evaluate the implemented system, I used the TAC10 dataset with the same parameters as those used in the previous chapter for a better performance comparison. Two other systems were used as baselines. The first utilizes the Trimmer algorithm and creates a single compressed version of each candidate sentence. The summary is then formed by aggregating the compressed versions of the highest ranking sentences. The second baseline is the summarizer implemented in the previous chapter using the strong weighted links without SSM. The results obtained with the ROUGE metric are illustrated in Table 6.1 and Figure 6.19. The letters A and B refer to the labels of the two main documents sets in TAC10, namely A and B which are both used during the testing stage for summarizing documents.

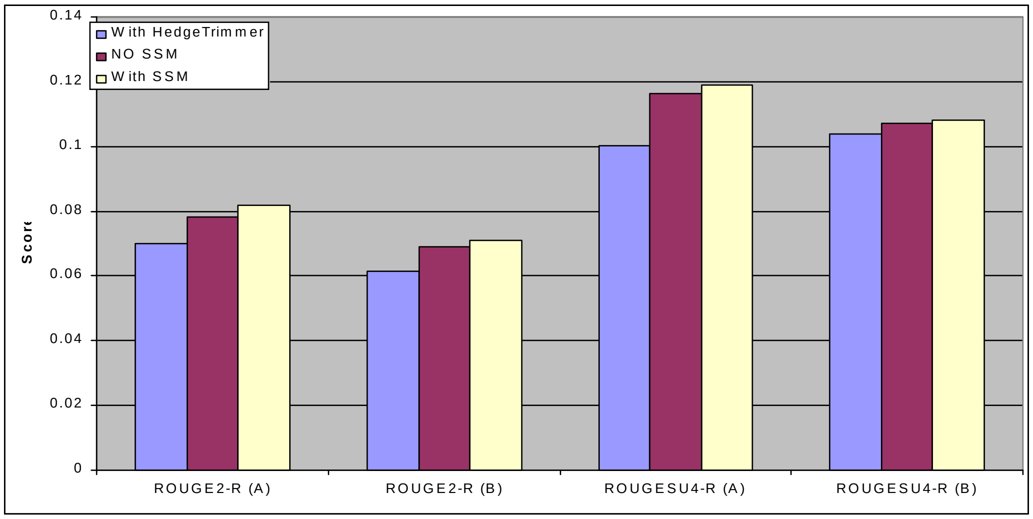


Figure 6.19: Comparison of the ROUGE results obtained for the different systems

Evaluation	Trimmer	NO SSM	With SSM
ROUGE2-R (A)	0.07011	0.07883	**0.08173**
ROUGE2-R (B)	0.06160	0.06901	**0.07101**
ROUGESU4-R (A)	0.10026	0.11889	**0.11917**
ROUGESU4-R (B)	0.10401	0.10702	**0.10815**

Table 6.1: ROUGE results suggest a performance improvement with the SSM-based summarizer

It can be noted from the results that the introduction of the SSM-based system led to various levels of improvements to the ROUGE results (1.1% to 4.6%) when compared against the original Wikipedia-based summarizer. This goes along with the intuition that compressing sentences should increase the capacity of a summary. With the increased capacity, it is vital to have a dynamic features selection that can aid with sentences selection. This is evident by examining the results of the Trimmer baseline where compressing all sentences in the summary as a post-processing stage caused a loss to the system's performance.

6.5 Conclusion

In this chapter, I described a module I implemented for simplifying sentences and producing multiple splits and compressed versions of each sentence. The simplification module, SSM, is aimed to help with summarization by segmenting sentences to remove non important parts while retaining relevant parts for inclusion in the summary. For this purpose, the syntactical interpretation of sentences allows for patterns detection and applying a set of rules to simplify sentences whenever possible. After obtaining multiple simplified versions of each sentence, another module within the Wikipedia-based summarizer chooses the most important and least redundant sentence to include in the summary. An evaluation was performed and the obtained and reported results indicate an achieved improvement in the summarizer.

Chapter 7

Classification Aided with Wikipedia

Documents classification is the task of assigning a document to one or more of previously defined categories based on the document content. After building the Features Generator for Wikipedia, I chose classification as the first application to explore. This was supported by the notion in [188] that text classification is the most suitable area to explore to test the effectiveness of newly constructed features. With documents classification, the system deals with documents previously constructed by humans with no user interaction. The results of the system classification can be checked for accuracy against the humans' classification for the same documents. In addition, no human queries that require further expansion or processing are usually needed with classification.

In this section, I describe a novel classification system that employs the Wikipedia extracted features in a unique way. Instead of mapping a documents text to a concept or a small group of concepts as done in most of the previous work, I map it to all of the previously-processed Wikipedia concepts. This is achieved by first processing all Wikipedia articles and extracting the relationship between each of its terms and all the concepts existing within Wikipedia as was described in section 5.3. This step is performed only once and its result is utilized during the training and classification stages to compute how similar each Wikipedia concept is to text fragments existing in the documents. In a following step, I employ the concepts hierarchies existing within Wikipedia to analyze the relationship between the concepts in each document. Furthermore, I apply a centroid-based method directly on the documents contents. The centroid-based classifier extracts inter-class, inner-document and inter-document features to form prototype vectors. The result of all the mentioned steps (Concepts, Categories and Text) is then combined to form a prototype vector for each class during the training stage. The classifier employs the formed vectors to decide which class each of the test documents belongs to in a fast and efficient way. My experimental results on the 20-newsgroup dataset and the ODP collection demonstrate that my classifier performs very well when compared to previous methods.

7.1 System Overview

I propose a system for classifying text documents with the aid of external knowledge. Figure 7.1 shows the main stages involved during the process. The system accepts as input the training documents along with the classes they belong to. In addition, the system takes as input the test document that needs to be classified. All of the documents are initially parsed and preprocessed to remove stop words and apply tokenization as was performed in the Wikipedia-based summarizer and described in section 5.5.1. Afterwards, the Wikipedia-extracted features are used to aid in enriching the provided training and test documents. This enrichment process produces several vectors reflecting different aspects of the documents including the important terms, top Exact Match (EM) concepts/related concepts, and the top M related categories. After generating the different vectors, they are passed to the similarity computing module which in return produces a sorted list of the top classes the Test Document TD is likely to belong to.

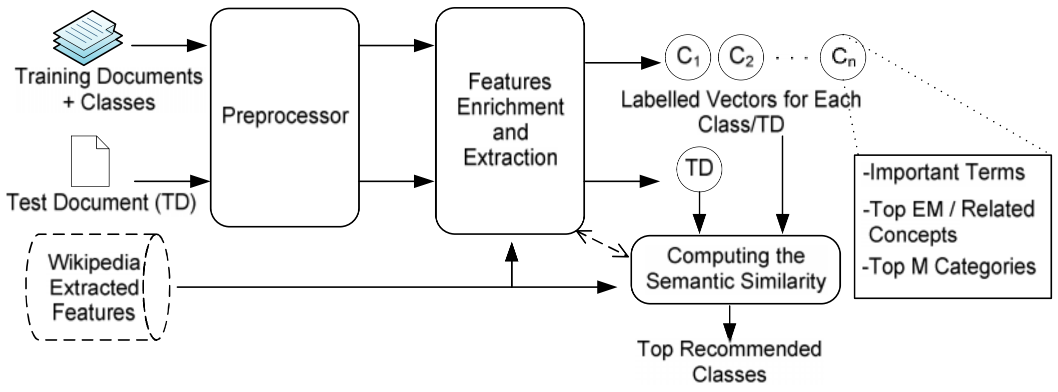

Training Documents + Classes
Test Document (TD)
Wikipedia Extracted Features
Preprocessor
Features Enrichment and Extraction
C1 C2 . . . Cn
Labelled Vectors for Each Class/TD
TD
-Important Terms
-Top EM / Related Concepts
-Top M Categories
Computing the Semantic Similarity
Top Recommended Classes

Figure 7.1: A general framework for Wikipedia-assisted Text Classification

The first stage in the framework is to preprocess all fed documents by parsing them first to extract the text and then tokenizing the terms. Following that is the removal of stop words. Terms distributions are then taken into account by computing the term frequency TF and Inverse Document Frequency IDF. The common TFIDF weighting scheme is used to find a weight for each term in the documents. These values are used in other system modules as described in the heuristics section 7.4.2.

In Figure 7.2, I show the class diagram of the implemented classifier with the main classes illustrated. The classes *textProcessor*, *DocumentPreprocessor* and *DocumentSet*, *Sentence*, *Term*, *WikiFeaturesRetrieval* and *Concept* are the same as those implemented in the WordNet-based summarization system and described in chapter 4 and 5. Each *Category* is for represented with *FeaturesVectors* which are built and prepared by the model *FeaturesBuilder*. When classifying documents, the similarity between *FeaturesVectors* is computed with help from the class *VectorsSimMeasure*.

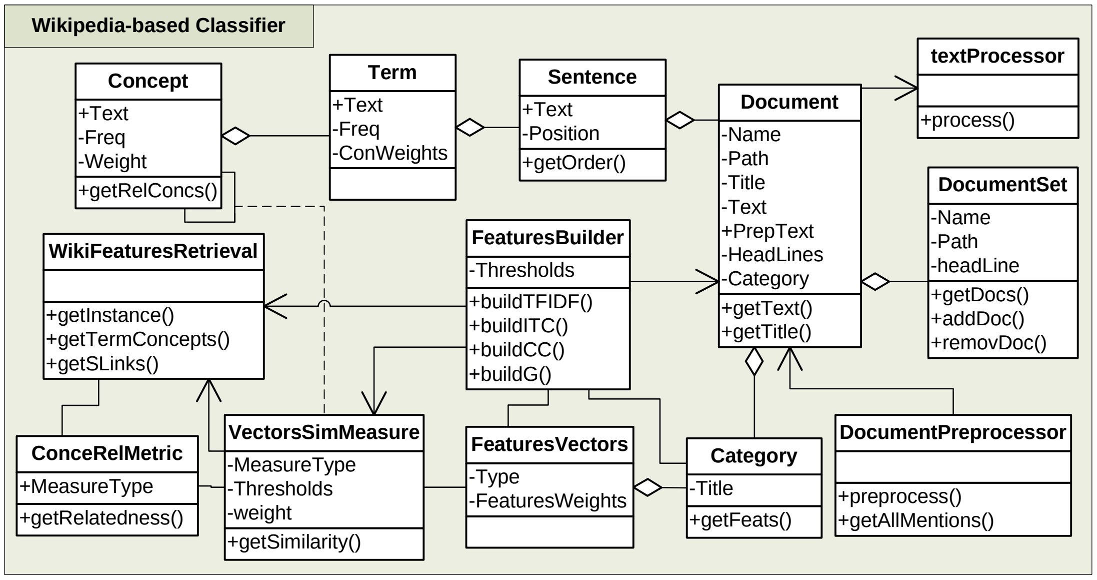


Figure 7.2 Class diagram of the *Wikipedia-based classifier* package

7.2 Using the Wikipedia-Extracted Features

After preprocessing the documents, I apply a set of heuristics focusing on different aspects of the input documents. The goal is to provide the top recommended classes for the Test Document TD with help from the best or all of these heuristics. For each class, I use the documents that belong to it as the basis for extracting representative features of that class. These features are represented as vectors in most of the applied heuristics and are evaluated according to the evaluation policy of the heuristic that generated the features. The scores of all evaluation vectors are aggregated at the end when computing the similarity between the classes' vectors and the TD vector. In the following sections, the extracted features are described along with the process applied to each.

7.2.1 Important Terms Extraction

For each class *C*, I have a group of training documents d_i belonging to that class, where *i* refers to the document number. The idea is to extract a list of the most important terms ordered by their weights to represent each class. The aim is to have representative terms for each class that best distinguish it from the rest of the text. This is similar to many other approaches previously implemented in different applications including documents clustering [165], cluster labelling [189] and automatic text summarization [43]. In my work, I chose to implement this and use it as one of the baselines during the evaluation comparisons.

I extract the important terms in each class and give each a score using a weight scheme similar to the one described in [189] and [190]. In the implemented metric, inter-class, inner-document and inter-document features are used to compute a weight for each term in each class. They are then grouped in a set for each class to form a representative centroid. For a term *t* in a class *C*, the term weight *w(t,C)* is derived using:

$$w(t,C) = ctf(t,C) \cdot \log(\frac{|C|}{CF_t}) \cdot idf(t,C) \qquad \textbf{7.1}$$

Where *ctf(t,C)* refers to the class term frequency and *idf(t,C)* is the inverse document frequency. The inner part of equation 7.1 is for the *inverse class frequency* and attempts to capture the different distributions of a term within the different classes. Intuitively, a term that appears in one class is deemed to be important for that class. Similarly, if a term appears in all of the classes, its discriminative power wouldn't be that important for text classification.

As for the TD, I simply find the TFIDF weight for each term. During evaluation, the formed TD vector is compared against the vectors of all classes using the cosine similarity measure. A ranked list of the recommended classes is then generated solely on the basis of the obtained similarity scores.

7.2.2 Concepts Extraction

The documents that belong to each class can be represented in the form of a concepts vector. The purpose here is to find the best representative concepts for each class that best separate the class's documents from the rest of the classes. This is usually achieved by devising a mapping scheme that uses the class documents contents to decide which concepts they are mostly related to. Previous approaches have been considered for achieving this. For example, [191] defined some Wikipedia relations between the articles such as synonymy and polysemy and used them with the categories structure to aid in mapping any text fragment to concepts. In [165], the authors created a Wikipedia concepts matrix from the content of the Wikipedia articles and used it as a bridge for the mapping. In my work, I employ different heuristics for this task utilizing the term-concepts table, the previously defined strong links and the concepts titles.

7.2.2.1 Expansion with Term-Concepts Table

In the first heuristic, I use the Term-Concepts table to form a vector of concepts for each document in a class. Afterwards, all of the concepts vectors in a class are aggregated and a centroid is created for each. When grouping the generated concepts for different adjacent terms, the aggregation helps boost the multi-word concepts describing the contexts of the targeted text. However, this is also not without a limitation as large segments of text usually produce noise. In addition, important concepts which are briefly mentioned in a relatively small number of sentences may not appear in the concepts list. I therefore treat each sentence as a separate text fragment and generate a concepts vector for each. I keep only a K maximum of the top representative concepts for each sentence before grouping them altogether in the document. This way, important concepts which are mentioned once or twice in the document would still appear in the document's concepts vector.

7.2.2.2 Expansion with Strong Links

In another heuristic I applied, I used a revised Exact Match (EM) approach that is most similar to the one suggested in [165] to find concepts in a document. The idea is to construct the concepts vector by finding the explicitly mentioned concepts within each document in a class. For a text fragment that covers more than one concept using the same terms, I only keep the concept spanning the largest number of terms. For example, each term in “cat fish”, namely *cat* and *fish*, is a title for a concept. However, because there is a third concept titled *Cat fish* that spans the two adjacent words, I only choose that concept.

After finding all of the EM concepts in a document and adding them to the list, I extend each of them with a maximum of *RL* of the most related concepts. I use the strong links method previously defined to find the most related concepts to an EM concept. The weight assigned to each EM concept is simply its frequency multiplied by a constant (chosen as 8 in my experiment). As for the weight of a related concept, it would be its frequency multiplied by the actual weight of the different links types given in Table 5.2 which depends on how strong the association is. This approach is enhanced with the redirect links to find alternative names or abbreviations for some concepts. The concepts lists for each document would then contain two types of concepts: EM concepts and their related concepts. When computing the similarity between a class concepts list and a TD concepts list, the cosine similarity measure is used.

When an ambiguous EM concept is encountered in a document, I attempt to find its exact meaning by using the Weighted Strong Links based methodology previously described but here taking into account the class label, the EM concept and its surrounding words.

7.2.3 Categories Mapping

A categories vector is generated for each class after forming the concepts vectors. The previously extracted categories structure from Wikipedia is utilized for this task. After obtaining the concepts vectors with a score associated to each concept, I form a new similar vector in which the concepts are replaced with the categories they belong to. When several concepts share the same parent category, the concepts scores are aggregated into one and then they are replaced with the shared parent category in the categories vector. This aggregation in turn signifies the dominant categories which are parents to the largest number of concepts in a class. The same process is applied to the TD as well to generate a categories vector.

7.3 Classifying a Test Document

In the previous sections, I described the different stages involved in obtaining the important terms from each class and enriching its documents content with Wikipedia-extracted features including concepts and categories vectors. I highlight here how to classify a TD and combine the previously defined similarity measures into one. This is achieved by computing the aggregated score *gs(TD, CL)* for the test document *TD* and the class *CL* as follows:

$$gs(TD,CL) = \sum_{i=1}^{q} \beta_i Sim_i(TD,CL) \qquad \textbf{7.2}$$

Where q is the total number of heuristics or methods I apply, i is the number of the heuristic and β is the weight I give to the different heuristics I implement. The sum of all β_i in the above formula is one and thus each β_i is always given a value between zero and one. In this work, I experimented with different values of β in the evaluation section as I deemed reasonable and reported their results.

7.4 Experiments and Evaluation

My approach relies on the use of the vast and highly organized human knowledge existing within Wikipedia giving it a major advantage over other approaches using smaller thesauruses such as WordNet or Open Directory Project (ODP). I use the preprocessed version of Wikipedia that was described in chapter 5 for the task at hand.

7.4.1 Dataset

I performed classification experiments on two data collections. The first is the 20 News Groups (20N) which has newsgroups documents that were categorized manually into 20 different classes. Each class has nearly 1,000 documents making the total number of documents available in the collection almost 20,000. The second collection was built from the ODP. I selected 100 random categories and downloaded 100 pages for each category. The total number of documents is 10,000. Both datasets are balanced in the sense that the number of contained documents in every category is almost the same.

To both collections, all documents are parsed and tokenized. For the 20N collection only the title, subject, keywords and content are kept. Also, I keep only the titles and the content of the downloaded pages for ODP. I computed the TFIDF weight for each term. Terms appearing in the titles and the keywords section have their weights doubled to emphasize their importance. Unless otherwise stated in one of the applied heuristics, the title and the keyword sections are treated as part of the content of the articles.

I divided each dataset into five random, but equivalently sized sub-datasets retaining the documents distribution balance in each category. I use the five sub-datasets to evaluate my methods five times. Then, I take the average of the obtained accuracies as the final evaluation result.

7.4.2 Methods and Evaluation Setup

I conducted several experiments to evaluate the different methods and heuristics suggested in this work. Specifically, I compared the performance of the following approaches:

- *TFIDF*: A centroid is formed for each class using the common TFIDF weights of its documents terms.
- *ITC*: Important Terms Centroid is created for each class based solely on the important terms in the class documents. Formula 7.1 is used to compute the terms weight in the centroid and TFIDF is used for the weights of TD. Cosine similarity is employed for comparisons.
- *CC-TM*: Concepts Centroid is created for each class. The concepts vector is constructed by expanding the documents terms with the term-concepts table as described in section 7.2.2.1.
- *CC-SL*: Concepts Centroid is constructed for each class with the help of the strong links after detecting the explicitly mentioned concepts in the document as described in section 7.2.2.2. All of the explicitly mentioned concepts within a document were expanded through the inclusion of their 10 mostly related concepts using the strong links method.
- *G*: Categories centroid is created for each class by aggregating the categories of the encountered concepts using the best *EM* method.
- *ITC*-CCTM: This forms vectors for the Important Terms in each class, and the Concepts using *CC-TM* method.
- *ITC*-CCSL: This forms two vectors for each class, namely Important Terms vector and EM Concepts extended with related concepts using the strong links method.

- *ITC*-G: Important terms, and categories vectors are created for each class. The categories vector is implemented using the EM method, though the concepts vectors are not used directly when computing the similarity between the different vectors.
- *ITC-CCTM-G*: Important Terms, Concepts and Categories vectors are created. The categories vector is implemented using the EM method while the concepts vector is constructed as in *CC-TM*.
- *ITC-CCSL-G*: Important Terms, Concepts and Categories vectors are created. This is similar to ITC-CCTM-G except that the concepts vector is constructed as in *CC-SL* instead.

For all the methods that utilize the term-concepts table, I applied the boosting to the table with the parameters *FLBS* and *SLBS* assigned to 1 and 1.05 respectively which I found to be reasonable at the time. When Forming the Concepts centroid, I chose 20 for *K* as the maximum number of concepts to be obtained for any term from the term-concepts table. Equation 7.2 takes part in the implementation of the approaches: *ITC-CCTM*, *ITC-CCSL*, *ITC-G*, *ITC-CCTM-G,* and *ITC-CCSL-G*. For these approaches different values of β has been assigned and the results I display are those obtained by the best combination. The evaluation accuracy for a run is measured by dividing the number of correctly categorized documents over the number of total documents in a class. As for the categories labels of the 20N, they were updated to include full descriptive names as shown in Table 7.1.

Original Label	Updated Label
sci.crypt	science cryptography
sci.med	science medicine
sci.space	science space
sci.electronics	science Electronics
rec.motorcycles	motorcycles
rec.sport.hockey	hockey
rec.sport.baseball	baseball
rec.autos	automobiles
soc.religion.christian	society religion christianity christian
talk.religion.misc	religion
misc.forsale	sale discount
comp.windows,x	computer windows
comp.graphics	computer graphics
comp.os.ms-windows.misc	computer os operating system Microsoft windows
comp.sys.ibm.pc.hardware	computer system ibm pc hardware
comp.sys.mac.hardware	computer system apple mac macintosh hardware
alt.athesim	atheism
talk.politics.gun	politics guns
talk.politics.mideast	politics mideast
talk.politics.misc	politics

Table 7.1: Updated labels for the different categories in 20N

7.4.3 Results

The results I obtained for the run experiments are shown in Table 7.2. *ITC-CCSL* achieved the best result on both datasets. For the 20N dataset, the β variable was assigned {0.3, 0.5, 0.2} for the Important Terms vector, Concepts vector and Categories vector, respectively. As for the ODP collection data, they were chosen to be {0.3 , 0.4, 0.4}. These values were chosen manually to provide the best performance for each variation on each datasets. On both collections, the run of *CC-SL*, which uses the strong links method to construct representative concepts vectors for documents yielded better results than *CC-TM*. However, they both achieved better results than the two baselines *TFIDF* and *ITC*. This suggests that the term-concepts table and the strong links methods can be both used to form good representative vectors of text documents in different NLP applications, including Text Summarization. The second best run on both datasets is ITC-CCSL which relies on the strong links method while TFIDF achieved the least accuracy in both runs.

The results obtained for *ITC-CCSL-G* and *ITC-CCTM-G* are only marginally better than those of *ITC-CCSL* and *ITC-CCTM*, respectively. Similarly, the effects of introducing the *G* measure on *ITC* are only marginal. This suggests that it may be possible to skip the implementation of the *G* run in other applications and still obtain comparable results to runs that have *G* implemented.

	20N	**ODP**
TFIDF	82.98	69.26
ITC	84.78	71.54
CC-TM	86.42	74.85
CC-SL	86.79	77.24
G	85.08	73.73
ITC-CCSL	87.66	78.03
ITC-CCTM	86.94	77.12
ITC-G	85.41	74.34
ITC-CCSL-G	**88.02**	**78.12**
ITC-CCTM-G	87.08	77.31

Table 7.2: Accuracy results obtained for different variations of the system

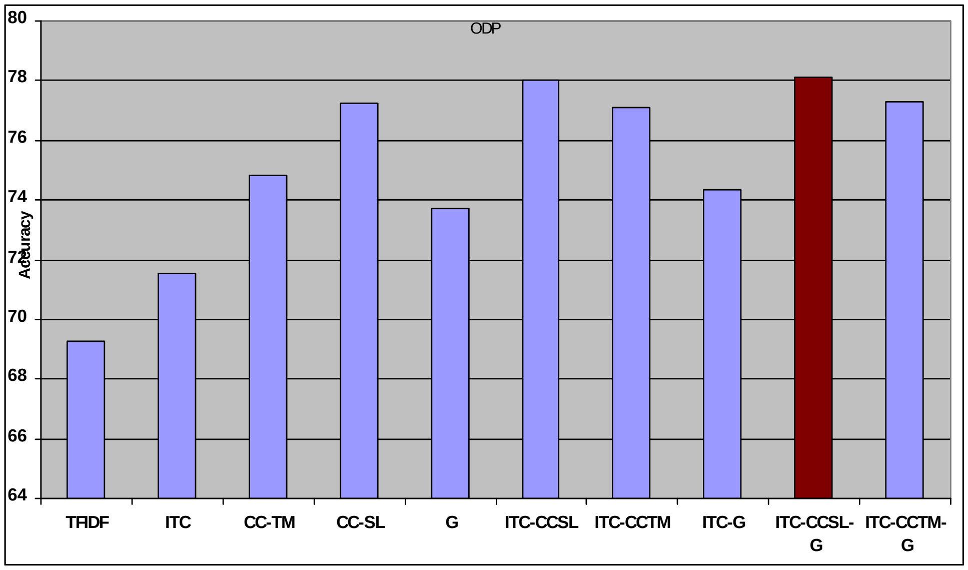


Figure 7.3: A comparison of the classification accuracies obtained for the different runs on the ODP dataset

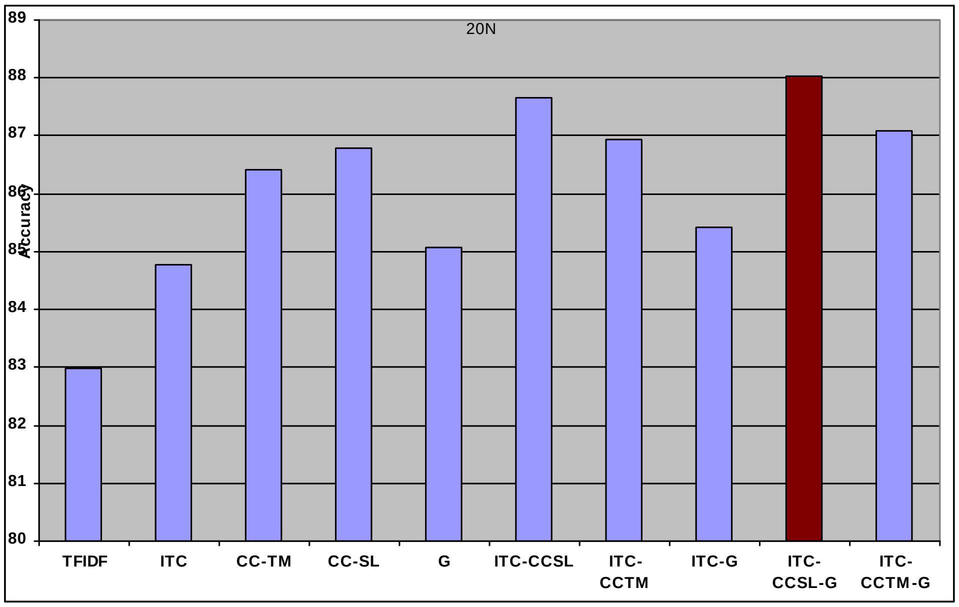


Figure 7.4A comparison of the classification accuracies obtained for the different evaluated runs on the 20N dataset

7.5 Conclusion

In this chapter, I used the features extracted from Wikipedia in the application of documents classification. I described the implementation of a text documents classifier and detailed its structure. I explained how the Wikipedia extracted features are used to aid in the processing stage of the classification system. To decide upon the effectiveness of the introduced features and how they compare against other common BOG metrics, I conducted an evaluation on two datasets with several runs for different methods. The obtained results indicate that the features constructed from Wikipedia can be better representative for test documents than other features such as TFIDF or ITC.

Chapter 8

Using SSM for Summarization

In this chapter, I illustrate how the SSM module works by providing sample sentences taken from the TAC10 dataset and their simplified versions as produced by SSM. I also include sample summaries generated by the SSM-enhanced summarizer which I described in chapter 6. In addition, I discuss the advantages and disadvantages of introducing SSM to summarization.

8.1 SSM for Simplifying Sentences

In section 6.2, I provided a description for SSM which aims to simplify sentences through applying a set of rules depending on the detected pattern and the parse tree of a sentence. SSM works by either splitting sentences into several versions or compressing sentences by removing several parts from it or both. In Figure 8.1 , I provide a summary of the rules applied by SSM. Rules 1 to 6 are for splitting sentences while rules 7 to 14 are for compressing sentences.

1	*SBAR , NP & < (WHNP << who) & << VP*
2	*SBAR , NP & < (WHNP << which) & << (VP , NP < VBD)*
3	*SBAR , NP & < (IN << that) & << (VP , NP < VBD)*
4	S < (NP . VP) , S < (NP.VP)
5	*VP , NP < (VP < VBD $ VP) < (CC < and | < or)*
6	*S < (NP . VP) $ (CC < but) $ S < (NP.VP)*
7	Keep Leftmost S Root and remove its siblings
8	Remove Time Expressions
9	Remove Conjunctions
10	Remove Complements
11	XP-over-XP
12	Remove PP under SBAR
13	Remove SBAR
14	Remove PP

Figure 8.1: Summary of the rules applied by SSM for simplifying sentences

I present here several examples illustrating the effect of applying SSM on different sentences.

The first example shows the effect of the removal of preposed adjuncts which is resulted from applying rule 7. Preposed adjuncts are defined in [24] as "*constituents that precede the first NP under the Root S*". The original sentence for this example is:

```
According to a previously undisclosed agreement between President
Barack Obama and his Mexican counterpart, Felipe Calderon, the
Pentagon is authorized to fly unmanned surveillance flights over
Mexico.
```

The parse tree of the above sentence is:

```
(ROOT
  (S
    (PP (VBG According)
      (PP (TO to)
        (NP
          (NP
            (NP (DT a) (RB previously) (JJ undisclosed) (NN agreement))
            (PP (IN between)
              (NP (NNP President) (NNP Barack) (NNP Obama))))
          (CC and)
          (NP
            (NP (PRP$ his) (JJ Mexican) (NN counterpart))
            (, ,)
            (NP (NNP Felipe) (NNP Calderon))))))
    (, ,)
    (NP (DT the) (NNP Pentagon))
    (VP (VBZ is)
      (VP (VBN authorized)
        (S
          (VP (TO to)
            (VP (VB fly)
              (NP (JJ unmanned) (NN surveillance) (NNS flights))
              (PP (IN over)
                (NP (NNP Mexico)))))))))
```

The simplified form of that sentence after applying rule 7 is:

```
The Pentagon is authorized to fly unmanned surveillance flights
over Mexico
```

The following sentence illustrates an example for the effects of applying the *NP-over-NP* and *PPs removal* rules. The original form of the sentence before simplification is:

```
Many highly-publicized and politically charged cases from the
trial of the officers charged with attacking Abner Louima to the
World Trade Center bombing  were not transferred to other venues.
```

The parse tree of the above sentence is:

```
(ROOT
  (S
    (NP
      (NP (JJ Many))
      (UCP
        (ADJP (RB highly) (VBN publicized))
        (CC and)
        (NP
          (NP
            (ADJP (RB politically) (VBN charged))
            (NNS cases))
          (PP (IN from)
            (NP
              (NP (DT the) (NN trial))
              (PP (IN of)
                (NP
                  (NP (DT the) (NNS officers))
                  (VP (VBN charged)
                    (PP (IN with)
                      (S
                        (VP (VBG attacking)
                          (NP
                            (NP (NNP Abner) (NNP Louima))
                            (PP (TO to)
                              (NP (DT the) (NNP World) (NNP Trade) (NNP
                              Center) (NN bombing))))))))))))))
    (VP (VBD were) (RB not)
      (VP (VBN transferred)
        (PP (TO to)
          (NP (JJ other) (NNS venues)))))))
```

After applying rules 11 and 14, the following simplified versions are obtained:

```
1- Many highly-publicized cases from the trial of the officers
   charged with attacking Abner Louima to the World Trade Center
   bombing were not transferred to other venues .
2- Many highly-publicized and politically-charged cases were not
   transferred to other venues.
```

The following example presents a sentence where a larger number of rules are applied.

The original sentence is:

```
The Rev Al Sharpton who organized the daily protests over the
Diallo shooting and is an adviser to the family also is expected
to attend the meeting, said Steven Reed, Johnson's spokesman.
```

The parse tree of the above sentence is:

```
(ROOT
  (SINV
    (S
      (NP
        (NP (DT The) (NNP Rev) (NNP Al) (NNP Sharpton))
        (SBAR
          (WHNP (WP who))
          (S
            (VP
              (VP (VBD organized)
                (NP (DT the) (JJ daily) (NNS protests))
                (PP (IN over)
                  (NP (DT the) (NNP Diallo) (NN shooting))))
              (CC and)
              (VP (VBZ is)
                (NP
                  (NP (DT an) (NN adviser))
                  (PP (TO to)
                    (NP (DT the) (NN family)))))))))
      (ADVP (RB also))
      (VP (VBZ is)
        (VP (VBN expected)
          (S
            (VP (TO to)
              (VP (VB attend)
                (NP (DT the) (NN meeting)))))))
    (, ,)
    (VP (VBD said))
    (NP
      (NP (NNP Steven) (NNP Reed))
      (, ,)
      (NP
        (NP (NNP Johnson) (POS 's))
        (NN spokesman)))))
```

When applying the rules individually and then jointly, SSM generated the following simplified sentences:

```
1- The Rev Al Sharpton who organized the daily protests over the
   Diallo shooting and is an adviser to the family is expected to
   attend the meeting.
2- Rev Al Sharpton who organized daily protests over Diallo
   shooting and is adviser to family is expected to attend meeting.
3- The Rev Al Sharpton also is expected to attend the meeting ,
   said Steven Reed , Johnson 's spokesman.
4- The Rev Al Sharpton organized the daily protests over the
   Diallo shooting and is an adviser to the family.
5- The Rev Al Sharpton organized the daily protests over the
   Diallo shooting
6- The Rev Al Sharpton is an adviser to the family.
7- Rev Al Sharpton is expected to attend meeting
```

The rules that were applied to generate the above sentences are 5, 11, 12, 13 and 14 individually and then jointly by having the output of one rule as the input for the other.

The following is yet another example illustrating the effects of another set of rules being applied. The original sentence is:

```
The accused officers are presumed innocent and deserve a fair and
impartial trial but there is no evidence that the people of the
Bronx and New York City cannot be trusted with this trial.
```

The parse tree of the above sentence is:

```
(ROOT
  (S
    (NP (NNP The))
    (VP (VBD accused)
      (SBAR
        (S
          (S
            (NP (NNS officers))
            (VP
              (VP (VBP are)
                (VP (VBN presumed)
                  (S
                    (ADJP (JJ innocent)))))
              (CC and)
              (VP (VBP deserve)
                (NP (DT a)
                  (ADJP (JJ fair)
                    (CC and)
                    (JJ impartial))
                  (NN trial)))))
          (CC but)
          (S
            (NP (EX there))
            (VP (VBZ is)
              (NP (DT no) (NN evidence))
              (SBAR (IN that)
                (S
                  (NP
                    (NP (DT the) (NNS people))
                    (PP (IN of)
                      (NP (DT the) (NNP Bronx)
                        (CC and)
                        (NNP New) (NNP York) (NNP City))))
                  (VP (MD can) (RB not)
                    (VP (VB be)
                      (VP (VBN trusted)
                        (PP (IN with)
                          (NP (DT this) (NN trial))))))))))))))
```

And the simplified versions of the above sentence are:

```
1- The accused officers are presumed innocent and deserve a fair
   and impartial trial but there is no evidence that the people
   can not be trusted.
2- The accused officers are presumed innocent and deserve a fair
   and impartial trial but there is no evidence .
3- There is no evidence that the people of the Bronx and New York
   City can not be trusted with this trial.
```

```
4- The accused officers are presumed innocent and deserve a fair
   and impartial trial.
5- The accused officers are presumed innocent.
6- The accused officers deserve a fair and impartial trial.
7- Accused officers are presumed innocent.
```

Which were the results of applying the rules 6, 7, 9, 11, 12, 13 and 14. In some cases, the parser produces inaccurate results as in the following example. The original sentence before simplification is:

```
Mrs Mandela, the former wife of Nelson Mandela, the anti-apartheid
hero and first black president of South Africa, compared police
brutality in New York to the oppression of black South Africans
under the system of segregation she fought to change.
```

The parse tree generated for the above sentence is:

```
(ROOT
  (S
    (NP
      (NP (NNP Mrs) (NNP Mandela))
      (, ,)
      (NP
        (NP (DT the) (JJ former) (NN wife))
        (PP (IN of)
          (NP
            (NP (NNP Nelson) (NNP Mandela))
            (, ,)
            (NP (DT the) (JJ anti-apartheid) (NN hero))
            (CC and)
            (NP (JJ first) (JJ black) (NN president))))
        (PP (IN of)
          (NP (NNP South) (NNP Africa))))
      (, ,))
    (VP (VBD compared)
      (NP (NN police) (NN brutality))
      (PP (IN in)
        (NP (NNP New) (NNP York)))
      (PP (TO to)
        (NP
          (NP (DT the) (NN oppression))
          (PP (IN of)
            (NP (JJ black) (NNP South) (NNPS Africans)))))
      (PP (IN under)
        (NP
          (NP (DT the) (NN system))
          (PP (IN of)
            (NP
              (NP (NN segregation))
              (SBAR
                (S
                  (NP (PRP she))
                  (VP (VBD fought)
                    (S
                      (VP (TO to)
                        (VP (VB change))))))))))))))
```

And the simplified versions of the above sentence as produced by SSM are:

```
1- Mrs Mandela , the former wife of Nelson Mandela , the anti-
   apartheid hero of South Africa , compared police brutality in
   New York to the oppression of black South Africans under the
   system of segregation she fought to change .
2- Mrs Mandela compared police brutality in New York to the
   oppression of black South Africans under the system of
   segregation she fought to change .
3- Mrs Mandela , the former wife of Nelson Mandela , the anti-
   apartheid hero and first black president of South Africa ,
   compared police brutality in New York to the oppression of
   black South Africans under the system of segregation.
4- Mrs Mandela , the former wife of Nelson Mandela , the anti-
   apartheid hero and first black president of South Africa ,
   compared police brutality in New York to the oppression of
   black South Africans .
5- Mrs Mandela , the former wife , compared police brutality .
6- Mrs Mandela compared police brutality in New York to oppression
   of black South Africans under system of segregation she fought
   to change
7- Mrs Mandela compared police brutality in New York to oppression
   of black South Africans under system of segregation
8- Mrs Mandela compared police brutality
```

The above-generated sentences were the results of applying the rules 5, 9, 11, 13 and 14 individually and then jointly by using the output of one rule as the input of another. As can be noted from the generated simplified sentences, sentences 5 and 8 appear to be incomplete. Sentence 5 was the output of applying rule 14 which removes all *PPs* from a sentence. Sentence 8 was generated as the final output of applying rules 13 and 14 jointly. This implies that the rules are not perfect and appear to generate incomplete or erratic sentences in some rare cases. In addition, the parser may produce incorrect parsing results for some complicated or irregular sentences as will be shown in the following section. Even with these limitations, using SSM to improve summarization can lead to a better performance as was highlighted in section 6.4. In the next section, I examine the output of the Wikipedia-based summarization system that was enhanced by SSM which was also described in section 6.3.

8.2 Summarizing with SSM

The goal of introducing SSM is to conserve as much space with the generated summaries as possible by keeping the important parts and removing the rest. In the following examples I show sample summaries generated by the strong links method of the Wikipedia-based summarizer which was described in section 6.3. The summaries are for different documents sets chosen from the TAC10 documents collection.

In Figure 8.2, I show two system summaries **A** and **B**. Summary A was generated without making use of SSM while summary B took advantage of SSM. By comparing summaries A and B, the difference can be noted in sentences 1 and 2 of summary A. For sentence 1, the temporal expression *on Saturday* was removed and resulted in sentence 1 of summary B. Sentence 2 on the other hand was simplified to the version shown in sentence 2 of summary B. The simplification of these two sentences allowed for expanding the total number of sentences in the generated summary from 4 as in summary A to 7 as shown in summary B. Furthermore, one of the newly added sentences in summary B was simplified before being added to summary B. The original form of sentence 5 in summary B appears in the test document as follows:

```
    Brian M Krzanich, vice president of the Technology and
    Manufacturing Group and general manager of Assembly/Test for Intel
    Corporation, said he felt proud Intel technological progress could
    be useful in protection of the giant pandas.
```

This was simplified to the following form before addition to summary B:

```
Brian M Krzanich said he felt proud Intel technological progress could
be useful in protection of giant pandas
```

Reference Summaries

1
China continues to move ahead in efforts to preserve the panda population. Seventy-eight mines and polluting companies, as well as three power companies, have been closed or suspended. A major conservation group has protested an offer of two pandas to Taiwan as a goodwill gesture. A museum is planned to showcase their protection efforts. A communications network is being established for information sharing. 161 pandas are in captive breeding programs, mostly in China, with insemination by both artificial and natural means. Even a medical facility has been set up for performing operations on injured pandas.

2
A regional telecom network which now covers China's Wolong Grant Panda Nature Reserve "will not only help increase the number of giant pandas but will also help us manage the living environment of giant pandas" the Reserve's director declared on April 3, 2005. On May 7 it was announced that China plans to build a new giant panda museum to help save the endangered animal and its habitat. On May 10 the Southwest Sichuan Province government announced that it had closed 78 mines and polluting companies in the giant panda's habitat.

3
A regional telecom network is helping to manage the endangered Great Panda and share information about it. Researchers at China's Wolong panda reserve can process panda data in real time, all day, and in all corners of the park. Other activities to protect the panda include the suspension of industrial expansion. Over 150 pandas are in captive breeding programs worldwide. Panda fertility is low, and offspring in zoos outside China must be returned to China when they mature, according to loan agreements. The panda is a CITES (Convention on International Trade in Endangered Species) protected animal; strict import-export laws apply.

4
Female giant pandas have a very short fertile period. Natural insemination in captivity is rare and, even with artificial insemination, births are rare and cubs are fragile. The Washington National and San Diego Zoos have had some success. China continues to pursue the World Natural Heritage site designation and closed mines and other polluters is the site area. China, France and the US are creating a giant panda museum in southwestern China. China also offered two pandas to Taiwan. China is beginning to use mobile technology at panda reserves and is using advanced surgical techniques to save injured pandas.

System Summary

A
(1) The pandas mated for about a half-hour on Saturday. **(2)** With the broadband network, researchers are able to process real-time data on the pandas, including photos and video signals, around the clock at any given corner of the nature reserve, or observe giant panda cubs on a daily basis without having to step out of their offices, according to Wu. **(3)** It's a giant panda cub no bigger than a stick of butter. **(4)** Artificial insemination resulted in the 1999 birth of Hua Mei, the first giant panda to survive more than four days in the U.S.

B
(1) The pandas mated for about a half-hour. **(2)** Researchers are able to process real-time data on pandas including photos around clock at any given corner of nature reserve. **(3)** It's a giant panda cub no bigger than a stick of butter. **(4)** Artificial insemination resulted in the 1999 birth of Hua Mei, the first giant panda to survive more than four days in the U.S. **(5)** Brian M Krzanich said he felt proud Intel technological progress could be useful in protection of giant pandas.**(6)** The area is home to about 300 wild giant pandas. **(7)** Panda gestation ranges from 90 to 185 days.

Figure 8.2: Reference Summaries and System Summaries for docset D1003B from the TAC10 documents collection

Reference Summaries

1
During the night of July 17, a 23-foot tsunami hit the north coast of Papua New Guinea (PNG), triggered by a 7.0 undersea earthquake in the area. At least four villages were hit, three completely destroyed. A government station, a Catholic mission, and a community school were destroyed. The death toll, officially 599 by July 19, is expected to rise considerably. Thousands are injured, hundreds missing, and up to 6,000 are homeless. The PNG Defense Force, police, health services, and the PNG Red Cross have been mobilized. Australia will transport relief supplies and provide a mobile hospital and medical personnel.

2
A tsunami ravaged the northern coast of Papua New Guinea during the night of 17 July 1998, when a magnitude 7.0 undersea earthquake caused a 23 foot wall of water to strike the West Sepik Province villages of Aitape, Nimas, Warapu, Arop, and Teles-Lambu. Among the area's approximate population of ten thousand, 599 are dead and 6000 homeless from beach abodes constructed of very fragile jungle materials. The Australian Air Force provided three C130 transports to ferry supplies to the devastated area. Australia also was sending a mobile hospital and doctors. Queen Elizabeth sent regrets.

3
On a Friday night in mid-July 1998, a 23-foot tsunami engulfed a heavily populated area near Aitape on Papua New Guinea's remote northwest coast. The tsunami followed 30 minutes after an undersea earthquake 18 miles off the coast, measuring 7.0 on the Richter scale. The death toll was 599 but could rise to 2,000. Hundreds were missing and up to 6000 could be homeless. Most of the dead were children and old people. Seven villages were completely destroyed, including Nimas, Warapu, and Arop. Three Australian C130s were bringing food, medical supplies and personnel and a mobile hospital.

4
A 23-feet tsunami struck the remote northwest coast of Papua New Guinea the night of Friday 17 July, totally destroying three villages and almost completely destroying another. The death toll, mostly children and old people, has reached 59 but is expected to rise to more than 1000 and there are thousands of injured and homeless, with no food or water. Australia is sending medical supplies and food and is expecting to set up a mobile hospital. The tidal wave was caused by an undersea earthquake measuring about 7.0 on the Richter scale some 12 miles off the coast.

System Summary

(1) A tsunami spawned by a 7 magnitude earthquake crashed into Papua New Guinea's north coast , crushing villages and leaving hundreds. **(2)** The death toll in Papua New Guinea's (PNG) tsunami disaster has climbed to 599 and is expected to rise. **(3)** The Papua New Guinea (PNG) Defense Force killing scores of people, on PNG's remote north-west coast Friday night. **(4)** Australia said it will provide transport for relief supplies and a mobile hospital to Papua New Guinea. **(5)** System would use seismological information from Australian Geological Survey Organization from NTF to predict where tsunami, tidal wave caused , would hit.

Figure 8.3: Reference Summaries and System Summaries for docset D1004A from the TAC10 documents collection

In another example illustrated in Figure 8.3 above, I have a system summary on which SSM was applied. As can be noted from sentences 1 and 5, it would appear that the generated sentences after being simplified are incomplete. The original form of sentence 1 that appeared in the test documents is as follows:

```
A tsunami spawned by a 7 magnitude earthquake crashed into Papua
New Guinea's north coast, crushing villages and leaving hundreds
missing, officials said Sunday.
```

Parsing that sentence would generate the following parse tree:

```
(ROOT
  (S
    (NP
      (NP (DT A) (NN tsunami))
      (VP (VBN spawned)
        (PP (IN by)
          (NP (DT a) (CD 7) (NN magnitude) (NN earthquake)))))
    (VP (VBD crashed)
      (PP (IN into)
        (S
          (S
            (NP
              (NP (NNP Papua) (NNP New) (NNP Guinea) (POS 's))
              (NN north) (NN coast)))
          (, ,)
          (S
            (VP
              (VP (VBG crushing)
                (NP (NNS villages)))
              (CC and)
              (VP (VBG leaving)
                (S
                  (NP (NNS hundreds))
                  (ADJP (VBG missing))))))
          (, ,)
          (S
            (NP (NNS officials))
            (VP (VBD said)
              (NP (NNP Sunday))))))))
```

When analyzing the generated parse, it would appear that *missing* does not belong to the tree subset of *crushing villages and leaving hundreds missing* but rather to the subset of *officials said Sunday*. This interpretation led to having the term *missing* removed along with *officials said Sunday* when applying rule 7 of the SSM module. The compressed sentence would then appear in the following form.

```
A tsunami spawned by a 7 magnitude earthquake crashed into Papua
New Guinea's north coast, crushing villages and leaving hundreds.
```

The incompleteness can also be noted in sentence 5 of summary B. The original sentence before compression appears in the following form:

```
The system would use seismological information from Australian
Geological Survey Organization and sea level data from NTF to
predict where a tsunami , a tidal wave caused by an earthquake ,
would hit and how much damage it would cause.
```

The parse tree of that sentence takes the following form:

```
(ROOT
  (S
    (NP (DT The) (NN system))
    (VP (MD would)
      (VP (VB use)
        (NP (JJ seismological) (NN information))
        (PP (IN from)
          (NP (JJ Australian) (NNP Geological) (NNP Survey) (NNP
Organization)
            (CC and)
            (NN sea) (NN level) (NNS data)))
        (PP (IN from)
          (NP (NNP NTF)))
        (S
          (VP (TO to)
            (VP (VB predict)
              (SBAR
                (SBAR
                  (WHADVP (WRB where))
                  (S
                    (NP
                      (NP (DT a) (NN tsunami))
                      (, ,)
                      (NP
                        (NP (DT a) (JJ tidal) (NN wave))
                        (VP (VBN caused)
                          (PP (IN by)
                            (NP (DT an) (NN earthquake)))))
                      (, ,))
                    (VP (MD would)
                      (VP (VB hit)))))
                (CC and)
                (SBAR
                  (WHNP (WRB how) (JJ much))
                  (S
                    (VP (VBP damage)
                      (SBAR
                        (S
                          (NP (PRP it))
                          (VP (MD would)
                            (VP (VB cause)))))))))))))))
```

After applying the SSM rules, I obtain the following sentences:

```
1- The system would use seismological information to predict where
   a tsunami, a tidal wave caused, would hit and how much damage
   it would cause
2- The system would use seismological information from Australian
   Geological Survey Organization and sea level data from NTF to
   predict
3- System would use seismological information from Australian
   Geological Survey Organization from NTF to predict where
   tsunami, tidal wave caused , would hit
4- System would use seismological information from Australian
   Geological Survey Organization from NTF to predict
5- system would use seismological information to predict
```

As can be noted above, all of the generated sentences appear to be incomplete. The original sentence defines the term *tsunami* as *a tidal wave caused by earthquake*. By looking at the parse tree, it can be noted that *by earthquake* is a PP under the SBAR *where a tsunami, a tidal wave caused by earthquake, would hit*. When applying any of the rules 12, 13 or 14, the phrase *by earthquake* would be removed. This is reflected in sentence 5 of the generated summary B in Figure 8.3.

8.3 Summary

In this chapter, I summarized the rules of SSM which, when applied to a sentence, aim to produce as many simplified versions of that sentence as possible. I then provided examples showing the effects of applying SSM to several sentences chosen from the TAC10 dataset. Afterwards, I provided two sets of example summaries: one for the Wikipedia-based summarizer and another for the Wikipedia-based summarizer that was enhanced with SSM and described in section 6.3. I also compared the generated summaries and highlighted the advantages and disadvantages of introducing SSM to the system. While it appears from the comparisons that SSM can sometimes produce erratic output in the form of incomplete sentences, the advantages it carries by generating complete sentences and preserving space within summaries outweigh its disadvantages in the task of summarization as was illustrated in the results of section 6.4.

Chapter 9

Summaries and Conclusion

9.1 Summary of Work

In this thesis, I proposed a feature generation methodology employing external knowledge bases for reasoning on textual documents. In order to apply a level of reasoning that is as close to that of humans as possible, I utilized two open-domain knowledge repositories crafted by humans to create features extractors and generators. The features extractors and generators analyze the text documents at hand, extract salient features and enrich their representation with new features with the collaboration of the external knowledge repositories. By relying only on what is available in text documents and not using the external repositories, it would not have been possible to enrich them with new features. Thus, external repositories would be useful in many Information Retrieval and Data Mining applications including Automatic Documents Summarization.

Several approaches were mentioned in the background section of this thesis with their limitations highlighted. Among the most common approaches is BOW. The use of external ontologies allows for bypassing the limitations of BOW methods by inferring knowledge about the terms that exist in a text document and defining the relationship between these terms. For example, terms that appear in the topic of a document may be different but related to what is mentioned in a section of the document. With external repositories, it may be possible to detect the relationship between the topic terms/concepts and the different sections of the text document. Two repositories have been used in my work: WordNet and Wikipedia.

With WordNet, algorithms for computing the semantic similarity between terms were proposed and implemented. The relationship between terms, and a composite of terms, is quantified and weighted through the new algorithms allowing for grouping the terms, phrases and sentences based on the semantic meaning they carry. These algorithms were especially useful when applied to the application of Automatic Documents Summarization as the evaluation results show. Several novel methods were also adapted to enhance the diversity and reduce redundancy in the generated summaries. The implemented methods utilize both the semantic relations and lexical links that exist within WordNet. As illustrated in the evaluations performed, the results obtained suggest a better performance of the summarization system when compared against others that do not employ the implemented redundancy-diversity methods.

As for Wikipedia, its use in the process of generating features for text fragments allowed for the introduction of human knowledge *Concepts.* Quantifying the relationship between the detected concepts with the methods I propose allow for a better Automatic Semantic Interpretation of Text. As the structure of Wikipedia was not designed to be machine readable as was the case with WordNet, preprocessing stages had to be applied first to extract its features. Each article in Wikipedia was viewed as a single *Concept*. The articles content, links and categories were used to help define the relatedness between the concepts. Two main sets of features were extracted from Wikipedia: Term-concepts table and Weighted Strong Links. When a text fragment is processed with the Wikipedia-assisted system, one of the two mentioned feature sets is applied. These two sets were also extended to deduce even a larger set of methodologies that can be applied to text fragments as described in Chapter 5.

I also applied the generated features to the problem of assessing the semantic relatedness between any two text fragments. With the Wikipedia extracted features, two methods were proposed. First is through the expansion of each term through the term-concepts table. Each term would be replaced with a vector of its most related concepts where a weight is assigned to each. This is followed by a merge applied to all the concepts vectors existing within a single text fragment. The merge in effect boosts the concepts which best represent the text fragment and degrades the rest. When comparing the two text fragments, applying a cosine measure to the two concepts vectors would quantify the relatedness between them. The second method is similar but relies on Weighted Strong Links instead of the term-concepts table when constructing the vectors.

The WordNet-based system and the Wikipedia-based framework were both adapted to be used in the application of Automatic Documents Summarization. To evaluate the WordNet-based system, I participated in the summarization task of the TAC08 conference. The system was evaluated and obtained an average rank in most of the evaluation measures. In TAC10, I participated again in the summarization task with the Wikipedia-assisted summarizer. Based on the evaluation results obtained, the system achieved a more competitive performance.

I investigated the use of a Sentences Simplifications Module (SSM) to generate shorter and simpler forms of sentences. Compressing and splitting sentences was applied in a step preceding the candidate sentences selection in the Wikipedia-based summarizer. SSM was used to generate multiple simplified versions of each sentence. The summary candidates' selection module included an iterative process that implicitly enforces redundancy checking while choosing simplified or un-simplified sentences to include in the summary based on the current summary state and the summary length required. The effect of introducing SSM and the iterative process was evaluated and the results show an overall improvement in the summarizer's performance.

In addition to Automatic Documents Summarization, the Wikipedia-assisted framework was adapted and used in two other applications, namely Word Sense Disambiguation and Automatic Documents Classification. Evaluations were also performed and the results suggest a competitive performance. It is possible to expand the work of the framework and apply it to other applications in the future including: documents clustering, clusters labelling and hierarchical documents classification.

9.2 Summary of the Evaluations Performed

The evaluations performed in this thesis can be classified into mainly two categories: Evaluations for measures and algorithms related to WordNet and evaluations related to Wikipedia.

9.2.1 WordNet-Related Evaluations

In Chapter 4, I described a set of algorithms detailing how the similarity between sentences can be computed. Namely, I proposed the following measures:

- arTonv_SemSimMeasure: This converts adjectives and adverbs to their corresponding nouns and verbs whenver possible before computing the semantic similarity between sentences.
- Syn_SimMeasure: When two sentences are compared, words in the second sentence are expanded with their synonyms.
- EditDist_SimMeasure: This computes the Levenshtein edit distance between terms when two sentences are compared.
- EditDistEx_SimMeasure: When computing the similarity between two sentences, words in the second sentences are expanded with their synonyms first. Afterwards,

the edit distances between terms from first sentence and second sentence are computed.

Details about the above mentioned measures with examples of their usages are provided in section 4.2.2.

Instead of relying only on the semantic relations that exist within WordNet, I also utilized WordNet's lexical relations for improving how the similarity between sentences is computed. This was especially evident with the proposed measure arTonv_SemSimMeasure described above. The evaluations performed on the TAC08 dataset for summarizing documents illustrate the performance increase with the new measure in the application of summarization. A summary of the results obtained is shown in Table 4.4. The results in the table also indicate that using semantic-based measures such as *Sem_SimMeasure* and *arTonv_SemSimMeasure* can lead to better system performance when compared against other non-semantic based methods.

When examining the generated summaries with the above measures, it can be noted that redundant sentences tend to appear in some summaries especially when producing a single summary for a large number of documents. This was shown in an example summary in Figure 4.23 which was generated for a document set from the TAC08 documents collection. In an attempt to reduce redundancy and increase diversity of summary sentences, I added a redundancy and diversity checking stage in the summarization system. It works by checking the redundancy or diversity of candidate sentences before they are added to the summary to ensure only non-redundant or diverse sentences are added to the summary. The measures that were used for implementing the redundancy checking stage are *Red-Syn*, *Red-Sim*, *Red-LevDist* and *Red-LevDist-Exp*. As for diversity checking, I considered using the measures named *Div-Ant* and *Div-Sim* which attempt to capture the diversity between sentences by checking the antonyms of the terms they contain. The results obtained after enforcing these measures in the redundancy and diversity checking stage indicate an overall performance increase as shown in Table 4.5. The only exception is the ROUGE1 result for the simple diversity checking measure Div-Ant which obtained a relatively inferior result. When examining the summaries obtained with the help of this measure, it was noticed that the detection of the wrong sense for a term lead to errors in generating its antonyms. An example for this is the term *feet* which was interpreted incorrectly as *part of the body* instead of *unit of length* leading to have *head* as its antonym.

The main contributions of the work performed in this section are:

- Considering both the lexical and semantic relations while computing the similarity between sentences
- Considering nouns, verbs, adjective and adverbs while computing the semantic similarity between sentences instead of relying only on nouns and verbs
- Introducing new measures for computing the similarity between sentence that lead to performance increase against traditional measures when used in summarization systems

9.2.2 Wikipedia-Related Evaluations

Because Wikipedia was not designed from the start to be machine-readable, a series of process had to be applied. First is extracting and constructing features from Wikipedia. For this, both the structure and content of Wikipedia were used. Second is devising means for augmenting text documents with the extracted features. Third, is interpreting these new features in the text document and deciding how to use them in the application at hand.

Two main sets of features were extracted from Wikipedia. One is called the *term-concepts table* and the other is *Strong links*. I described in Chapter 5 how they are used to represent text documents. I also explained how they can be used to compute the semantic relatedness between sentences and text fragments in section 5.5.4.

After I developed a module for extracting features from Wikipedia and proposed several methodologies for how the features can be mapped to text documents and then interpretted, it was necessary to evaluate the effectiveness of the features and the mapping and interpretation processes. The first chosen application for this was *documents classification* as no user input is usually required during the different evaluation stages and the evaluation is usually more robust as all documents have already been manually classified by humans. I conducted several experiments on the ODP and 20N datasets to evaluate the different heuristics suggested in my work. The results of the evaluations are shown in Table 7.2. They suggest that the usage of the Wikipedia-extracted features does indeed lead to performance improvements in the task of classification when compared against other BOW-based methods.

I also used the extracted features in the task of WSD. This was necessary for testing the best weights to choose for the different links types defined in the *strong links* method. In addition, it was useful to use as a component in the summarization system to help in disambiguating terms. The results of the evaluation results obtained are illustrated in Table 5.4 and indicate better performance for the strong links method when compared agains the rest.

I then used the term-concepts table and strong links for building an automatic summarizer. To evaluate the summarizer, I used the TAC10 dataset and compared it against two other baselines. The results obtained indicate competitive performance for the methods implemented in the new system. The *strong links* method achieved better performance than the other utilizing the *term-concepts table*. The evaluation results for this are shown in Table 5.6. When examining sample summaries generated by each method, it was observed that usage of the right context size has a significant effect on the performance of the term-concepts methods. Choosing a large number leads to a *noise* by having a large number of weakly related concepts detected as top concepts. On the other hand, choosing a small number affects the ability of the system in disambiguating concepts spanning multiple terms. An example summary illustrating this was provided in section 5.5.7. This notion was also supported by the results of the WSD evaluation in section 5.4.4.

In an attempt to enhance the performance of the Wikipedia-based summarizer even further, I implemented SSM which aims to provide multiple simplified versions of any processed sentence. This was supported by the notion that the top ranking sentences detected by the summarizer tend to be long and thus occupy a large space within summaries. An example illustrating this was provided in section 8.2. I provided details of how SSM was implemented and performed an evaluation testing its effect on the built summarizer. The obtained results with the ROUGE metric shown in Table 6.1 indicate an improvement in the performance of the summarizer.

In Chapter 8 I provided examples showing the effect of SSM on a sample of sentences chosen from the TAC10 documents collection. I also illustrated with an example summary how SSM affected the summarizer. It was observed during the examination of the obtained output from SSM that not all sentences are complete and that SSM is not error-free. This can be caused by many reasons. The parser used for parsing sentences and the simplification rules applied may provide incorrect or incomplete sentences for sentences having grammatical errors or uncommon syntactical structures. However, as was noted in the evaluation performed in Chapter 6 and the analysis in section 8.2, the advantages of using SSM outweigh its disadvantages.

9.3 Future Work

Several parameters had to be assigned manually while running the different summarizers described in this thesis. These parameters include the redundancy threshold, α (which decides how much weight given to user query or documents titles), context size (for the term-concepts table method) and summary limit. It would be optimal to have at least some of these parameters automatically chosen by the system. A module that can aid in determining this based on the length or the genre of the original text documents may be useful. The same also applies to the context size when implementing methods or algorithms involving the term-concepts table.

The evaluations performed in this thesis indicate that performance of summarization systems in general can be improved through the usage of external knowledge repositories. For the system to perform even better, the genre of the processed documents and query nature may need to be identified. Documents with different genres may need to be handled differently. For example, consider having a large number of movie reviews that need to be summarized. Different aspects of the document would have to be handled differently including the plot, user sentiment and rating. Consider the case of question answering. Having asked a specific questions such as “when”, “who” or “where”, the system would have to deal with each of these questions differently. Further investigation on this topic and on how to make the extracted features useful for handling specific genres of documents is needed.

In my future work, I plan to use the built and extracted features in several other applications such information search, documents clustering and clusters labelling. With information search, it may be useful to augment the query and the documents with human knowledge concepts for better text interpretation and retrieval. Indexing documents would also factor in the concepts contained within the documents in addition to their original terms. For documents clustering, the methodologies that were applied to documents classification in this thesis may need to be adapted. The discovery of the main topics within documents and clustering based on the dominant concepts may yield good performing clustering systems. The availability of the vast knowledge within the built features, the concepts titles that were crafted by humans should also be useful for clusters labelling.

I anticipate extending the implemented methods by applying new algorithms for matching text documents with concepts and investigating better techniques for representing the attributes of each concept. In this work, the focus for Wikipedia main features was on its main links, articles titles, articles content text content and categories. Other useful information was not considered such as the different versions of its pages in different languages. An article written in Spanish, for example, may include some links to related articles which do not exist in its English counterpart. These links may be especially useful to include as extra features for defining the relatedness between concepts, especially in the application of Word Sense Disambiguation.

In addition to WordNet and Wikipedia, I plan to use other open-world repositories such as Wiktionary. Wiktionary is a freely available multilingual web-based dictionary. It shows many similarities with expert-made repositories such as WordNet in that it contains concepts which are connected by lexical semantic relations and described with a gloss giving a short definition to the concept. Its size exceeds the size of WordNet. However, its openness, incompleteness and structure make it difficult to extract its features and the process is prone to parsing errors.

9.4 Conclusion

In this thesis, I presented and evaluated several algorithms for computing the similarity between sentences and text fragments through utilizing WordNet’s lexical and semantic relations. I also described methodologies detailing how features can be extracted from the content and structure of Wikipedia making it machine readable. The extracted features from Wikipedia have been used to augment the representation of text documents and compute the semantic relatedness between any two text fragments. Through the methodologies developed and algorithms implemented, I utilized both WordNet and Wikipedia in the application of Automatic Documents Summarization. The evaluations performed show that the usage of external knowledge repositories allows for improvements in the performance of systems in the task of summarization.

REFERENCES

[1] H. Luhn, “The Automatic Creation of Literature Abstracts,” *IBM Journal of Research and Development*, vol. 2, no. 2, 1958.

[2] K. S. Jones, “Automatic Summarising: Factors and Directions,” *Advances in Automatic Text Summarization*, pp. 1-12, 1998.

[3] D. Lenat, “From 2001 to 2001: Common sense and the mind of HAL,” in *Hal’s Legacy*, MIT Press, 1997, pp. 194-209.

[4] I. Mani, D. House, G. Klein, L. Hirschman, T. Firmin, and B. Sundheim, “The TIPSTER SUMMAC Text Summarization Evaluation,” in *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, Stroudsburg, PA, USA, 1999, pp. 77–85.

[5] R. Tucker, “Automatic summarising and the CLASP system,” University of Cambridge Computer Laboratory, 1999.

[6] D. R. Radev, E. Hovy, and K. McKeown, “Introduction to the special issue on summarization,” *Computational Linguistics*, vol. 28, pp. 399–408, Dec. 2002.

[7] D. Das and A. F. T. Martins, “A Survey on Automatic Text Summarization,” *Literature Survey for the Language and Statistics II course at CMU*, Nov. 2007.

[8] P. Baxendale, “Man-made index for technical literature - an experiment,” *I.B.M. Journal of Research and Development*, vol. 2, no. 4, 1958.

[9] R. Brandow, K. Mitze, and L. F. Rau, “Automatic condensation of electronic publications by sentence selection,” *Information Processing & Management*, vol. 31, no. 5, pp. 675-685, Sep. 1995.

[10] J. Kupiec, J. Pedersen, and F. Chen, “A Trainable Document Summarizer,” *SIGIR ’95 Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, p. 68--73, 1995.

[11] E. Hovy and C.-Y. Lin, “Automated text summarization and the SUMMARIST system,” in *Proceedings of a workshop on held at Baltimore, Maryland: October 13-15, 1998*, Stroudsburg, PA, USA, 1998, pp. 197–214.

[12] G. Salton, *Automatic text processing: the transformation, analysis and retrieval of information by computer*. Reading, Mass: Addison-Wesley Longman Publishing Co., Inc., 1989.

[13] H. P. Edmundson, “New Methods in Automatic Extracting,” *Journal of the ACM (JACM)*, vol. 16, pp. 264–285, Apr. 1969.

[14] C. Orasan, V. Pekar, and L. Hasler, “A comparison of summarisation methods based on term specificity estimation,” in *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC2004)*, 2004.

[15] S. Teufel and M. Moens, “Sentence Extraction as a Classification Task,” *Proceedings of the ACL/EACL-97 Workshop on Intelligent Scalable Text Summarization*, p. 58--65, 1997.

[16] J. J. Pollock and A. Zamora, “Automatic Abstracting Research at Chemical Abstracts Service,” *Journal of Chemical Information and Computer Sciences*, vol. 15, no. 4, pp. 226-232, Nov. 1975.

[17] C. Aone, M. E. Okurowski, and J. Gorlinsky, “Trainable, scalable summarization using robust NLP and machine learning,” in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, Stroudsburg, PA, USA, 1998, pp. 62–66.

[18] D. Cristea, O. Postolache,, G. Puscasu, and L. Ghetu, “Summarizing documents based on cue-phrases and references,” in *Proceedings of the International Symposium on Reference Resolution and Its Applications to Question Answering and Summarization*, Venic, Italy, 2003.

[19] M. Jaoua and A. B. Hamadou, “Automatic Text Summarization of Scientific Articles Based on Classification of Extract’s Population,” in *Computational Linguistics and Intelligent Text Processing*, vol. 2588, A. Gelbukh, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 623-634.

[20] X. Wan, J. Yang, and J. Xiao, “Towards an Iterative Reinforcement Approach for Simultaneous Document Summarization and Keyword Extraction,” *Proceedings of ACL2007*, 2007.

[21] J. M. Conroy and D. P. O’leary, “Text summarization via hidden Markov models,” in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2001, pp. 406–407.

[22] H. Daume III and D. Marcu, “Bayesian Multi-Document Summarization at MSE,” *Workshop on Multilingual Summarization Evaluation (MSE)*, 2005.

[23] J. Fürnkranz, T. Mitchell, and E. Riloff, “A Case Study in Using Linguistic Phrases for Text Categorization on the WWW,” *In Working Notes of the AAAI/ICML Workshop on Learning for Text Categorization*, p. 5--12, 1998.

[24] B. Dorr, D. Zajic, and R. Schwartz, “Hedge Trimmer: A Parse-and-Trim Approach to Headline Generation,” in *Proceedings of HLT-NAACL 2003 Text Summarization Workshop*, 2003, vol. 5.

[25] C.-yew Lin, “Training a Selection Function for Extraction,” in *Proceedings of the Eighteenth Annual International ACM Conference on Information and Knowledge Management (CIKM)*, Kansas City, 1999, pp. 55-62.

[26] R. Jin, M. Abu-Ata, Y. Xiang, and N. Ruan, “Effective and efficient itemset pattern summarization: regression-based approaches,” in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, 2008, pp. 399–407.

[27] Y.-L. Chen and L. T.-H. Hung, “Using decision trees to summarize associative classification rules,” *Expert Systems with Applications: An International Journal*, vol. 36, pp. 2338–2351, Mar. 2009.

[28] R. M. Schwartz, T. Imai, F. Kubala, L. Nguyen, and J. Makhoul, “A maximum likelihood model for topic classification of broadcast news,” in *EUROSPEECH*, 1997.

[29] K. Knight and D. Marcu, “Statistics-Based Summarization - Step One: Sentence Compression,” in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, 2000, pp. 703–710.

[30] K. Knight and D. Marcu, “Summarization beyond sentence extraction: a probabilistic approach to sentence compression,” *Artificial Intelligence*, vol. 139, pp. 91–107, Jul. 2002.

[31] J. Turner and E. Charniak, “Supervised and unsupervised learning for sentence compression,” in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Stroudsburg, PA, USA, 2005, pp. 290–297.

[32] D. Zajic, B. Dorr, and R. Schwartz, “Automatic Headline Generation for Newspaper Stories,” *In the Proceedings of the ACL Workshop on Automatic Summarization/Document Understanding Conference (DUC)*, p. 78--85, 2002.

[33] J. Conroy and S. Judith, “CLASSY Query-Based Multi-Document Summarization,” in *Proceedings of the 2005 Document Understanding Conference (DUC-2005) at*

NLT/EMNLP 2005, Vancouver, Canada, 2005.
[34] M. Osborne, “Using maximum entropy for sentence extraction,” in *Proceedings of the ACL-02 Workshop on Automatic Summarization - Volume 4*, Stroudsburg, PA, USA, 2002, pp. 1–8.
[35] V. Hatzivassiloglou, J. L. Klavans, M. L. Holcombe, R. Barzilay, M.-yen Kan, and K. R. McKeown, “SIMFINDER: A Flexible Clustering Tool for Summarization,” *In Proceedings of the NAACL Workshop on Automatic Summarization*, p. 41--49, 2001.
[36] I. Titov and R. Mcdonald, “A Joint Model of Text and Aspect Ratings for Sentiment Summarization,” *PROC. ACL-08: HLT*, p. 308--316, 2008.
[37] K. Svore, L. Vanderwende, and C. Burges, “Enhancing Single-Document Summarization by Combining RankNet and Third-Party Sources,” in *Proceedings of EMNLP-CoNLL*, 2007, pp. 448-457.
[38] C. Burges, T. Shaked, E. Renshaw, M. Deeds, N. Hamilton, and G. Hullender, “Learning to rank using gradient descent,” *IN ICML*, p. 89--96, 2005.
[39] G. B. Orr and K.-R. Müller, *Neural Networks: Tricks of the Trade*, 1st ed. Springer, 1999.
[40] K. Kaikhah, “Automatic Text Summarization with Neural Networks,” in *Proceedings of second international Conference on intelligent systems, IEEE*, Texas, USA, 2004, pp. 40-44.
[41] S. P. Yong, A. I. Z. Abidin, and Y. Y. Chen, “A neural-based text summarization system,” in *Data Mining VII: Data, Text and Web Mining and their Business Applications*, Prague, Czech Republic, 2006, pp. 185-192.
[42] L. Hasler, “From Extracts to Abstracts: Human summary production operations for computer-aided summarisation,” University of Wolverhampton, 2007.
[43] A. Bawakid and M. Oussalah, “A Semantic Summarization System: University of Birmingham at TAC 2008,” in *Proceedings of the First Text Analysis Conference*, 2008.
[44] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, “Five Papers on WordNet,” in *WordNet: An Electronic Lexical Database*, MIT Press, 1998.
[45] H. Jing and K. Mckeown, “Cut and paste based text summarization,” in *Proceedings of the Sixth Applied Natural Language Conference (ANLP-00) and the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-00)*, 2000, pp. 178-185.
[46] B. Boguraev and C. Kennedy, “Salience-based Content Characterisation of Text Documents,” *Advances in Automatic Text Summarization*, p. 2--9, 1997.
[47] J. Jiang and D. Conrath, “Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy,” in *Proceedings on International Conference on Research in Computational Linguistics*, Taiwan, 1997, pp. 19-33.
[48] D. Lin, “An Information-Theoretic Definition of Similarity,” *In Proceedings of the 15th International Conference on Machine Learning*, p. 296--304, 1998.
[49] J. R. Curran, “From Distributional to Semantic Similarity,” PhD Thesis, School of Informatics, University of Edinburgh, 2003.
[50] D. Wang, T. Li, S. Zhu, and C. Ding, “Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization,” in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2008, pp. 307–314.
[51] R. Barzilay and M. Elhadad, “Using lexical chains for text summarization,” in *In Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, 1997, pp. 10-17.
[52] M.-yen Kan, K. R. McKeown, and J. L. Klavans, “Domain-Specific Informative and

Indicative Summarization for Information Retrieval," *In Workshop on Text Summarization (DUC 2001)*, vol. 78, p. 1629--1636, 2001.

[53] K. R. McKeown, J. L. Klavans, V. Hatzivassiloglou, R. Barzilay, and E. Eskin, "Towards multidocument summarization by reformulation: progress and prospects," in *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, Menlo Park, CA, USA, 1999, pp. 453–460.

[54] B. Baldwin and T. S. Morton, "Dynamic Coreference-Based Summarization," *In Proceedings of The Third conference on Empirical Methods in Natural Language Processing (EMNLP-3)*, 1998.

[55] U. Hahn and U. Reimer, "Knowledge-based Text Summarization: Salience and generalization Operators for Knowledge Base Abstraction," in *Advances in Automatic Text Summarization, Mani & Maybury (Ed.)*, 1999.

[56] E. Lloret, O. Ferrández, R. Muñoz, and M. Palomar, "A Text Summarization Approach Under the Influence of Textual Entailment," in *Proceedings of the 5th International Workshop on Natural Language Processing and Cognitive Science (NLPCS 2008)*, 2008, pp. 22-31.

[57] M. A. K. Halliday and P. R. Hasan, *Cohesion in English*, 1st ed. Longman, 1976.

[58] A. Correira, "Computing story trees," *Computational Linguistics*, vol. 6, pp. 135–149, Jul. 1980.

[59] K. Ono, K. Sumita, and S. Miike, "Abstract generation based on rhetorical structure extraction," in *Proceedings of the 15th conference on Computational linguistics - Volume 1*, Stroudsburg, PA, USA, 1994, pp. 344–348.

[60] W. Mann and S. Thompson, "Rhetorical structure theory: Toward a functional theory of text organization," University of Southern California / Information Sciences Institute, Technical Report RR-87-190, 1988.

[61] D. Marcu, "Improving Summarization Through Rhetorical Parsing Tuning," in *Proceedings of the Workshop on Very Large Corpora*, Montreal, Canada, 1998.

[62] D. Marcu, "The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts," University of Toronto, Toronto, Ontario, Canada, 1998.

[63] L. Alonso i Alemany and M. Fuentes Fort, "Integrating cohesion and coherence for automatic summarization," in *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 2*, Stroudsburg, PA, USA, 2003, pp. 1–8.

[64] I. Cunha, S. Fernández, P. Velázquez Morales, J. Vivaldi, E. SanJuan, and J. M. Torres-Moreno, "A New Hybrid Summarizer Based on Vector Space Model, Statistical Physics and Linguistics," in *MICAI 2007: Advances in Artificial Intelligence*, vol. 4827, A. Gelbukh and Á. F. Kuri Morales, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 872-882.

[65] D. R. Radev and K. R. McKeown, "Generating natural language summaries from multiple on-line sources," *Computational Linguistics*, vol. 24, pp. 470–500, Sep. 1998.

[66] H. Jing and K. R. McKeown, "The decomposition of human-written summary sentences," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 1999, pp. 129–136.

[67] G. Grefenstette, "Producing Intelligent Telegraphic Text Reduction to Provide an Audio Scanning Service for the Blind," *Intelligent Text Summarization*, no. Intelligent Text Summarization, pp. 111-117, 1998.

[68] I. Mani, *Automatic Summarization*. John Benjamins Pub Co, 2001.

[69] E. Reiter and R. Dale, *Building Natural Language Generation Systems*. Cambridge University Press, 2000.

[70] K. McKeown, J. Robin, and K. Kukich, "Generating concise natural language summaries," *Information Processing and Management: an International Journal*, vol. 31, pp. 703–733, Sep. 1995.

[71] F. Liu and Y. Liu, "From extractive to abstractive meeting summaries: can it be done by sentence compression?," in *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, Stroudsburg, PA, USA, 2009, pp. 261–264.

[72] S. Guiasu and A. Shenitzer, "The principle of maximum entropy," *The Mathematical Intelligencer*, vol. 7, no. 1, pp. 42-48, Mar. 1985.

[73] G. Carenini and J. C. K. Cheung, "Extractive vs. NLG-based abstractive summarization of evaluative text: the effect of corpus controversiality," in *Proceedings of the Fifth International Natural Language Generation Conference,* Stroudsburg, PA, USA, 2008, pp. 33–41.

[74] J. Carbonell and J. Goldstein, "The use of MMR, diversity-based reranking for reordering documents and producing summaries," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 1998, pp. 335–336.

[75] J. Goldstein, V. Mittal, J. Carbonell, and J. Callan, "Creating and evaluating multi-document sentence extract summaries," in *Proceedings of the ninth international conference on Information and knowledge management*, New York, NY, USA, 2000, pp. 165–172.

[76] F. Boudin, J.-M. Torres-Moreno, and M. El-Bèze, "Improving Update Summarization by Revisiting the MMR Criterion," in *Proceedings of CoRR*, 2010.

[77] R. Ribeiro and D. M. de Matos, "Extractive summarization of broadcast news: comparing strategies for European portuguese," in *Proceedings of the 10th international conference on Text, speech and dialogue*, Berlin, Heidelberg, 2007, pp. 115–122.

[78] Y. Li and B. Merialdo, "Multi-video summarization based on Video-MMR," in *International Workshop on Image Analysis for Multimedia Interactive Services*, Desenzano del Garda, Italy, 2010.

[79] R. Mihalcea and P. Tarau, "TextRank: Bringing Order into Texts," in *Conference on Empirical Methods in Natural Language Processing*, 2004.

[80] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," in *Computer Networks and ISDN Systems*, Amsterdam, The Netherlands, The Netherlands, 1998, vol. 30, pp. 107–117.

[81] G. Salton, A. Singhal, M. Mitra, and C. Buckley, "Automatic text structuring and summarization," *Information Processing and Management: an International Journal*, vol. 33, pp. 193–207, Mar. 1997.

[82] I. Mani and E. Bloedorn, "Summarizing Similarities and Differences Among Related Documents," *Information Retrieval*, vol. 1, pp. 35–67, May 1999.

[83] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, p. 391--407, 1990.

[84] W. Song and S. C. Park, "A Novel Document Clustering Model Based on Latent Semantic Analysis," in *Third International Conference on Semantics, Knowledge and Grid (SKG 2007),* Xi'an, Shan Xi, China, 2007, pp. 539-542.

[85] B. Yu, Z.-ben Xu, and C.-hua Li, "Latent semantic analysis for text categorization using neural network," *Knowledge-Based Systems*, vol. 21, pp. 900–904, Dec. 2008.

[86] J.-Y. Yeh, H.-R. Ke, W.-P. Yang, and I.-H. Meng, "Text summarization using a

trainable summarizer and latent semantic analysis," *Information Processing and Management: an International Journal*, vol. 41, pp. 75–95, Jan. 2005.

[87] J. Steinberger, M. A. Kabadjov, and M. Poesio, "Improving LSA-based summarization with anaphora resolution," in *In Proceedings of the Human Language Technology Conference And Conference on Empirical Methods in Natural Language Processing*, 2005, p. 1--8.

[88] J. Steinberger and K. Ježek, "Using latent semantic analysis in text summarization and summary evaluation," *IN PROC. ISIM '04*, vol. 4, p. 93--100, 2004.

[89] J. D. Brown, "Developing an automatic document classification system: A review of current literature and future directions," Defence R&D Canada - Ottawa, Ottawa ONT (CAN), Technical Report, Jan. 2010.

[90] W. Kraaij, M. Spitters, and A. Hulth, "Headline Extraction Based on a Combination of Uni- and Multidocument Summarization Techniques," in *Proceedings of the ACL workshop on Automatic Summarization/Document Understanding Conference (DUC 2002)*, 2002.

[91] T. Euler, "Tailoring Text Using Topic Words: Selection and Compression," in *In IEEE Computer Society Press (ED.), Proceedings of the 3rd International Workshop on Natural Language and Information Systems (NLIS)*, 2002, p. 215--219.

[92] D. Z. Bonnie and B. Dorr, "BBN/UMD at DUC-2004: Topiary," *In Proceedings of the 2004 Document Understanding Conference (DUC 2004)*, p. 112--119, 2004.

[93] L. Zhou and E. Hovy, "Headline summarization at ISI," in *Proceedings of the Document Understanding Conference*, Edmonton, Alberta, Canada, 2003.

[94] F. Lacatusu et al., "GISTexter at DUC 2006: Multi-Strategy Multi-Document Summarization," in *2006 Document Understanding Conference (DUC 2006)*, 2006.

[95] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM COMPUTING SURVEYS*, vol. 38, no. 1, p. 2, 2006.

[96] M. E. J. Newman, "The Structure and Function of Complex Networks," *SIAM Review*, vol. 45, no. 2, pp. 167-256, 2003.

[97] W. Wang et al., "GraphMiner: a structural pattern-mining system for large disk-based graph databases and its applications," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, New York, NY, USA, 2005, pp. 879–881.

[98] K. Zechner, "Automatic Summarization of Spoken Dialogues in Unrestricted Domains," PhD Thesis, Carnegie Mellon University, School of Computer Science, Language Technologies Institute, Pittsburgh, PA 15213, 2001.

[99] I. Gurevych and M. Strube, "Semantic similarity applied to spoken dialogue summarization," in *Proceedings of the 20th international conference on Computational Linguistics*, Stroudsburg, PA, USA, 2004.

[100] S. Furui, T. Kikuchi, Y. Shinnaka, and C. Hori, "Speech-to-Text and Speech-to-Speech Summarization of Spontaneous Speech," *IEEE Trans. on Speech and Audio Processing*, vol. 12, no. 4, p. 401--408, 2004.

[101] H. Saggion, D. Radev, S. Teufel, W. Lam, and S. M. Strassel, "Developing Infrastructure for the Evaluation of Single and Multi-document Summarization Systems in a Cross-lingual Environment," *IN LREC 2002, PAGES 747–754, LAS PALMAS, GRAN CANARIA*, p. 747--754, 2002.

[102] D. R. Radev, H. Jing, and M. Budzikowska, "Centroid-Based Summarization of Multiple Documents: Sentence Extraction, Utility-Based Evaluation, and User Studies," in *Proceedings of the ANLP/NAACL 2000 Workshop on Automatic Summarization*, 2000.

[103] D. Radev, S. Blair-Goldensohn, and Z. Zhang, "Experiments in Single and Multi-Document Summarization Using MEAD," in *In First Document Understanding Conference*, New Orleans, LA, 2001.

[104] D. R. Radev, W. Fan, and Z. Zhang, "WebInEssence: a personalized web-based multi-document summarization and recommendation system," *In NAACL 2001 Workshop on Automatic Summarization*, p. 79--88, 2001.

[105] K. R. McKeown et al., "Tracking and Summarizing News on a Daily Basis with Columbia's Newsblaster," in *In Proceedings of Human Language Technology Conference (HLT 2002)*, 2002.

[106] R. Barzilay, K. R. McKeown, and M. Elhadad, "Information Fusion in the Context of Multi-Document Summarization," in *In Proceedings of the 37th Annual Meeting of the ACL*, 1999, p. 550--557.

[107] D. M. Dunlavy, D. P. O'Leary, J. M. Conroy, and J. D. Schlesinger, "QCS: A system for querying, clustering and summarizing documents," *Information Processing & Management*, vol. 43, no. 6, pp. 1588-1605, Nov. 2007.

[108] N. Madnani, D. Zajic, B. Dorr, N. Fazil Ayan, and J. Lin, "Multiple Alternative Sentence Compressions for Automatic Text Summarization," in *Proceedings of the 2007 Document Understanding Conference*, 2007.

[109] C. Sauper, A. Haghighi, and R. Barzilay, "Incorporating content structure into text analysis applications," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Stroudsburg, PA, USA, 2010, pp. 377–387.

[110] D. Zajic, B. Dorr, J. Lin, and C. Monz, "A sentence-trimming approach to multidocument summarization," *In Proceedings of the Document Understanging Conference 2005 (DUC-2005)*, pp. 151-158, 2005.

[111] L. Zhou and E. Hovy, "A web-trained extraction summarization system," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, Stroudsburg, PA, USA, 2003, pp. 205–211.

[112] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, vol. 41, no. 6, pp. 391-407, 1990.

[113] J. Steinberger and K. Ježek, "Text Summarization and Singular Value Decomposition," in *Advances in Information Systems*, vol. 3261, Springer Berlin / Heidelberg, 2005, pp. 245-254.

[114] N. Cristianini, J. Shawe-Taylor, and H. Lodhi, "Latent Semantic Kernels," *Journal of Intelligent Information Systems*, vol. 18, pp. 127–152, Mar. 2002.

[115] T. Liu, Z. Chen, B. Zhang, W.-ying Ma, and G. Wu, "Improving Text Classification using Local Latent Semantic Indexing," in *Data Mining, IEEE International Conference on*, Los Alamitos, CA, USA, 2004, vol. 0, pp. 162-169.

[116] H. Jing, R. Barzilay, K. Mckeown, and M. Elhadad, "Summarization Evaluation Methods: Experiments and Analysis," in *In AAAI Symposium on Intelligent Summarization*, 1998, p. 60--68.

[117] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Stroudsburg, PA, USA, 2002, pp. 311–318.

[118] C. H. And, C. Hori, and T. Hori, "Evaluation Method for Automatic Speech Summarization," in *Evaluation Method for Automatic Speech Summarization*, Geneva, Switzerland, 2003.

[119] C.-yew Lin, "Rouge: a package for automatic evaluation of summaries," in

Proceedings of the Workshop on Text Summarization Branches Out, post-conference workshop of ACL 2004, Barcelona, Spain, 2004, p. 25--26.

[120] A. Nenkova and R. Passonneau, “Evaluating Content Selection in Summarization: The Pyramid Method,” in *Proceedings of the 2004 Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics Annual Meeting*, 2004.

[121] H. Saggion, S. S. Dp, E. Uk, D. Radev, S. Teufel, and W. Lam, “Meta-evaluation of Summaries in a Cross-lingual Environment Using Content-based Metrics,” *IN PROCEEDINGS OF COLING-2002*, p. 849--855, 2002.

[122] K. Zechner and A. Waibel, “Minimizing word error rate in textual summaries of spoken language,” in *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, San Francisco, CA, USA, 2000, pp. 186–193.

[123] K. S. Jones and J. R. Galliers, “Evaluating natural language processing systems: an analysis and review,” in *Communications of the ACM*, 1996, vol. 39, pp. 73-80.

[124] T. E. Fawcett, “Feature discovery for problem solving systems,” PhD Thesis, University of Massachusetts, Amherst, MA, USA, 1993.

[125] S. Mohammad, “Measuring Semantic Distance using Distributional Profiles of Concepts,” PhD Thesis, University of Toronto, 2008.

[126] S. Nirenburg, S. Beale, and M. McShane, “Evaluating the performance of the OntoSem semantic analyzer,” in *Proceedings of the 2nd Workshop on Text Meaning and Interpretation*, Barcelona, Spain, 2004, pp. 33-40.

[127] *ACM Computing Classification System.* 1998.

[128] D. B. Lenat, R. V. Guha, K. Pittman, D. Pratt, and M. Shepherd, “Cyc: toward programs with common sense,” *Communications of the ACM*, vol. 33, pp. 30-49, Aug. 1990.

[129] The Editors of Time-Life Books, “Understanding computers: Artificial intelligence,” in *Understanding computers: Artificial intelligence*, Time-Life books Inc., 1986, p. 84.

[130] D. Milne, O. Medelyan, and I. H. Witten, “Mining Domain-Specific Thesauri from Wikipedia: A Case Study,” in *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, 2006, pp. 442-448.

[131] C. Fellbaum, *WordNet: an electronic lexical database*. Cambridge Mass: MIT Press, 1998.

[132] G. A. Miller, C. Leacock, R. Tengi, and R. T. Bunker, “A Semantic Concordance,” in *Proceedings of the workshop on Human Language Technology*, Stroudsburg, PA, USA, 1993, pp. 303–308.

[133] R. Rada, H. Mili, E. Bicknell, and M. Blettner, “Development and application of a metric on semantic nets,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 1, pp. 17-30, 1989.

[134] Shen Wan and R. A. Angryk, “Measuring semantic similarity using wordnet-based context vectors,” in *2007 IEEE International Conference on Systems, Man and Cybernetics*, Montreal, QC, Canada, 2007, pp. 908-913.

[135] R. Richardson and A. F. Smeaton, “Using wordnet in a knowledge-based approach to information retrieval,” School of Computer Applications, Dublin City University, Ireland, Working Paper CA-0395, 1995.

[136] P. Resnik, “Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language,” *Journal of Artificial Intelligence Research*, vol. 11, p. 95--130, 1999.

[137] G. Varelas, E. Voutsakis, E. G. M. Petrakis, E. E. Milios, and P. Raftopoulou, “Semantic Similarity Methods in WordNet and their Application to Information

Retrieval on the Web," *In the 7th ACM International Workshop on Web Information and Data Management (WIDM 2005)*, pp. 10-16, 2005.

[138] D. Lenat and R. Guha, *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[139] E. Gabrilovich and S. Markovitch, "Overcoming the brittleness bottleneck using Wikipedia: enhancing text categorization with encyclopedic knowledge," in *Twenty-First AAAI Conference on Artificial Intelligence*, 2006.

[140] J. Giles, "Internet encyclopaedias go head to head," *Nature*, vol. 438, no. 7070, pp. 900-901, Dec. 2005.

[141] K. A. Clauson, H. H. Polen, M. N. K. Boulos, and J. H. Dzenowagis, "Scope, completeness, and accuracy of drug information in Wikipedia," in *Annals of Pharmacotherapy*, 2008, vol. 42, pp. 1814-1821.

[142] D. Fallis, "Toward an epistemology of Wikipedia," *Journal of the American Society for Information Science and Technology*, vol. 59, pp. 1662–1674, Aug. 2008.

[143] H. Cunningham, H. Cunningham, R. J. Gaizauskas, and R. J. Gaizauskas, "A General Architecture for Text Engineering (GATE)," *Computers and the Humanities*, no. 36, pp. 223-254, 2002.

[144] H. Cunningham, "JAPE: a Java Annotation Patterns Engine," Department of Computer Science, University of Sheffield, Research Memorandum CS–99–06, 2000.

[145] H. Cunningham and D. Scott, "Software Architecture for Language Engineering," *Natural Language Engineering*, vol. 10, pp. 205–209, Sep. 2004.

[146] T. Pedersen and S. Patwardhan, "Wordnet::similarity - measuring the relatedness of concepts," in *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, San Jose, CA, 2004, p. 1024--1025.

[147] M. Greenwood, *Pure Java WordNet Similarity Library*. 2007.

[148] M. F. Porter, "An algorithm for suffix stripping," in *Program*, San Francisco, CA, USA, 1980, vol. 14, pp. 130-137.

[149] C. Nobata, S. Sekine, M. Murata, U. Kiyotaka, M. Utiyama, and H. Isahara, "Sentence Extraction with Information Extraction technique," in *Proceedings of the Second NTCIR Workshop Meeting*, 2001, pp. 213-218.

[150] M. Bautin and S. Skiena, "Concordance-Based Entity-Oriented Search," in *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*, Fremont, CA, USA, 2007, pp. 586-592.

[151] V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Sov. Phys. Dokl. 10 Original in Russian in Dokl. Akad. Nauk SSSR 163*, 1966, pp. 707-710.

[152] E. Hovy, C.-yew Lin, and L. Zhou, "Evaluating DUC 2005 using Basic Elements," *Proceedings of DUC2005*, 2005.

[153] A. Bawakid and M. Oussalah, "A Semantic-Based Text Classification System," presented at the 8th IEEE International Conference on Cybernetic Intelligent Systems, 2009, p. 135.

[154] G. Giannakopoulos, V. Karkaletsis, and G. Vouros, "Testing the use of N-gram Graphs in Summarization Sub-tasks," in *Proceedings of the First Text Analysis Conference*, 2008.

[155] R. Verma, D. Kent, and P. Chen, "Semantic Multi-document Update Summarization Techniques," in *Proceedings of the First Text Analysis Conference*, 2008.

[156] P.-E. Genest, G. Lapalme, L. Nerima, and E. Wehrli, "A Symbolic Summarizer for the Update Task of TAC 2008," in *Proceedings of the First Text Analysis Conference*, 2008.

[157] E. Wehrli, “Fips, a ‘deep’ linguistic multilingual parser,” in *Proceedings of the Workshop on Deep Linguistic Processing*, Stroudsburg, PA, USA, 2007, pp. 120–127.

[158] D. Galanis and P. Malakasiotis, “AUEB at TAC 2008,” in *Proceedings of the First Text Analysis Conference*, 2008.

[159] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273-297, Sep. 1995.

[160] T. Mansuy and R. J. Hilderman, “A characterization of wordnet features in Boolean models for text classification,” in *Proceedings of the fifth Australasian conference on Data mining and analystics - Volume 61*, Darlinghurst, Australia, Australia, 2006, pp. 103–109.

[161] M. de B. Rodriguez, J. M. G. Hidalgo, and B. D. Agudo, “Using WordNet to Complement Training Information in Text Categorization,” in *Proceedings of the Second International Conference on Recent Advances in Natural Language Processing*, 1997.

[162] K. Dave, S. Lawrence, and D. M. Pennock, “Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews,” in *WWW-2003*, 2003, p. 519--528.

[163] D. Milne, “Computing Semantic Relatedness using Wikipedia Link Structure,” in *Proceedings of the New Zealand Computer Science Research Student Conference*, 2007.

[164] E. Gabrilovich and S. Markovitch, “Computing semantic relatedness using Wikipedia-based explicit semantic analysis,” *In Proceedings of the 20th International Joint Conference On Artificial Intelligence*, p. 1606--1611, 2007.

[165] X. Hu, X. Zhang, C. Lu, E. K. Park, and X. Zhou, “Exploiting Wikipedia as external knowledge for document clustering,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Paris, France, 2009, pp. 389-396.

[166] R. Bunescu and M. Pasca, “Using Encyclopedic Knowledge for Named Entity Disambiguation,” in *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06), Trento, Italy*, 2006, pp. 9-16.

[167] S. P. Ponzetto and M. Strube, “Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution,” *IN PROC. OF HLT/NAACL*, p. 192--199, 2006.

[168] R. Mihalcea, “Using Wikipedia for Automatic Word Sense Disambiguation,” in *North American Chapter of the Association for Computational Linguistics (NAACL 2007)*, 2007.

[169] R. Mihalcea, “Turning WordNet into an Information Retrieval Resource: Systematic Polysemy and Conversion to Hierarchical Codes.,” *IJPRAI*, pp. 689-704, 2003.

[170] D. Turdakov and P. Velikhov, “Semantic Relatedness Metric for Wikipedia Concepts Based on Link Analysis and its Application to Word Sense Disambiguation,” in *SYRCoDIS*, 2008, vol. 355.

[171] R. Mihalcea and A. Csomai, “Wikify!: linking documents to encyclopedic knowledge,” in *CIKM ’07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, Lisbon, Portugal, 2007, pp. 242, 233.

[172] A. Fogarolli, “Word Sense Disambiguation Based on Wikipedia Link Structure,” in *Semantic Computing, 2009. ICSC ’09. IEEE International Conference on*, 2009, pp. 77-82.

[173] C.-yew Lin and E. Hovy, “From Single to Multi-document Summarization: A Prototype System and its Evaluation,” in *In Proceedings of the ACL*, 2002, vol. 48, pp. 457-464.

[174] C.-Y. Lin and E. Hovy, “Automatic evaluation of summaries using N-gram co-occurrence statistics,” in *Proceedings of 2003 Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Canada, 2003, vol. 1, pp. 71-78.

[175] J. P. Gee, *An Introduction to Discourse Analysis: Theory and Method*, 1st ed. Routledge, 1999.

[176] A. Siddharthan, “Syntactic Simplification and Text Cohesion,” *Research on Language and Computation*, vol. 4, no. 1, pp. 77-109, Mar. 2006.

[177] J. Carroll, G. Minnen, Y. Canning, S. Devlin, and J. Tait, “Practical Simplification of English Newspaper Text to Assist Aphasic Readers,” in *In Proc. of AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, 1998, pp. 7-10.

[178] J. Conroy, J. Schlesinger, D. Schlesinger, and J. Goldstein, “Back to Basics: CLASSY 2006,” in *Proceedings of the Sixth Document Understanding Conference (DUC)*, 2006.

[179] J. Conroy, J. Schlesinger, and D. Schlesinger, “CLASSY 2007 at DUC 2007,” in *Proceedings of the Seventh Document Understanding Conference (DUC)*, Rochester, New York, 2007.

[180] S. Blair-Goldensohn et al., “Columbia University at DUC 2004,” in *In 4th Document Understanding Conference (DUC 2004) at HLT/NAACL*, 2004.

[181] H. Jing, “Sentence Reduction for Automatic Text Summarization,” *In Proceedings of the 6th Applied Natural Language Processing Conference*, p. 310--315, 2000.

[182] H. Liu, Q. Zhao, Y. Xiong, L. Li, and C. Yuan, “The CIST Summarization Systems at TAC 2010,” in *Proceedings of the Text Analysis Conference (TAC)*, Gaithersburg, Maryland USA, 2010.

[183] C.-Y. Lin, “Improving summarization performance by sentence compression: a pilot study,” in *Proceedings of the sixth international workshop on Information retrieval with Asian languages - Volume 11*, Stroudsburg, PA, USA, 2003, pp. 1–8.

[184] D. Zajic, B. Dorr, J. Lin, and C. Monz, “A sentence-trimming approach to multidocument summarization,” in *In Proceedings of the Document Understanging Conference 2005 (DUC-2005)*, 2005, p. 151--158.

[185] L. Vanderwende, H. Suzuki, C. Brockett, and A. Nenkova, “Abstract Beyond SumBasic: Task-Focused Summarization with Sentence Simplification and Lexical Expansion,” *Information Processing and Management*, vol. 43, no. 6, pp. 1606-1618, 2007.

[186] E. Ong, J. Damay, G. Lojico, K. Lu, and D. Tarantan, “Simplifying Text in Medical Literature,” *Journal of Research in Science, Computing and Engineering*, vol. 4, no. 1, 2008.

[187] S. Jonnalagadda and G. Gonzalez, “Sentence Simplification Aids Protein-Protein Interaction Extraction,” in *Proceedings of the 3rd International Symposium on Languages in Biology and Medicine*, 2009, pp. 109-114.

[188] D. D. Lewis, “An evaluation of phrasal and clustered representations on a text categorization task,” in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 1992, pp. 37–50.

[189] D. Carmel, H. Roitman, and N. Zwerdling, “Enhancing cluster labeling using wikipedia,” in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, Boston, MA, USA, 2009, pp. 139-146.

[190] H. Guan, J. Zhou, and M. Guo, “A class-feature-centroid classifier for text categorization,” in *Proceedings of the 18th international conference on World wide*

web, Madrid, Spain, 2009, pp. 201-210.

[191] J. Hu et al., “Enhancing text clustering by leveraging Wikipedia semantics,” in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, Singapore, Singapore, 2008, pp. 179-186.

Index

APPENDICES

APPENDIX A

Sample Summaries Generated by the Implemented Systems

I provide here sample summaries generated by the best runs of my WordNet and Wikipedia summarizers. I start by providing sample single document summaries generated by the best run of my WordNet summarizer which was described and evaluated in section 4.3. The original document I consider is named `LTW_ENG_20050302.0121` and is part of the D0818D-A document set in the TAC08 documents collection. The document is shown in Figure A.1 and Figure A.2 in its original form.

```
<DOC id="LTW_ENG_20050302.0121" type="story" >
<HEADLINE>
Authorities Probe Evidence as Judges Call for Better Protection
</HEADLINE>
<TEXT>
A window shard, spent shell casings and a bloody mop emerged as key pieces of
evidence Wednesday in the apparent execution-style murders of the husband and
mother of a U.S. district court judge in Chicago.

Returning to her home on the city's north side Monday, Judge Joan Humphrey
Lefkow discovered the bodies of her husband, Michael Lefkow, 64, and mother,
Donna Humphrey, 89. Both victims had been shot to death.

Authorities stressed that federal agents and local detectives had not yet
narrowed the scope of their investigation. But several law-enforcement
officials and media reports Wednesday indicated that task force members were
leaning toward considering it a premeditated crime. The victims apparently
were forced to lie down on the basement floor before they were shot in the
head and chest.

``There is nothing spur-of-the moment or anything that would indicate this was
a crime of passion,'' one official said, adding that the slayings bore the
hallmarks of ``an execution.''

Authorities also found a bloody shoe print and a blood-streaked mop,
indicating there was an effort to clean up the crime scene, the Chicago
Tribune reported.
```

Figure A.1: Part 1 of the document named `LTW_ENG_20050302.0121` which was taken from the document set D0818D-A in the TAC08 dataset

```
Chicago police spokesman David Bayless said FBI forensics experts at the
bureau's laboratory in Quantico, Va., would analyze ballistics evidence
retrieved from the home. At least two .22-caliber shell casings reportedly
were recovered from the Lefkow residence.

Bayless and other officials would not provide specific details about
evidence retrieved from the house, out of concern that leaks about the
investigation might aid the killer or killers. ``We're going to be vague
and general for the time being,'' he said.

A security detail of federal marshals was guarding Lefkow and other family
members at an undisclosed location. The judge and her husband briefly
received protection last year during the trial of a white supremacist who
later was convicted of trying to arrange her murder.

The extremist leader, Matthew Hale, is awaiting sentencing next month.

In an interview Wednesday for the Chicago Tribune, Lefkow said she always
knew her job put her at risk but never thought it would endanger her
family.

``It's so unthinkable,'' the judge said. ``I imagine my husband must have
just walked into something,'' Lefkow said. ``Both of them were on crutches.
They didn't have a chance.''

Two U.S. district judges who work with Lefkow called for officials
Wednesday to reassess security arrangements for the nation's federal
judiciary. ``This horrible tragedy has got to serve as the basis for a
substantial increase in security for judges and their families,'' U.S.
District Judge Wayne R. Andersen said. ``The Internet is plastered with
information about every one of us, and I fear -- and my family certainly
fears -- that these kinds of incidents are going to be repeated unless
there is a very high priority on the safety of judges and their families.''

U.S. District Judge Marvin E. Aspen also asked for a new look at security
measures. Dan Lehman, a spokesman for the U.S. Northern District of
Illinois, said that Andersen had talked to Chief Judge Charles P. Kocoras
about his concerns and that some other judges working out of Chicago's
downtown federal courthouse had expressed worries since the killings.

``There's real concern that there needs to be a dispassionate,
comprehensive look at security arrangements in light of these homicides,''
Lehman said. The judge's husband had recently injured his leg and had
surgery.

</TEXT>
</DOC>
```

Figure A.2: Part 2 of the document named `LTW_ENG_20050302.0121` which was taken from the document set D0818D-A in the TAC08 dataset

It can be noted that the document contains a headline which I treat as the Topic of the document. This is represented as ***T*** in the scoring formula detailed in section 4.2.2.3. The document set which this document belongs to also has a title which is *`Judge Joan Lefkow's Family Murdered`*. I treat this title as a user query (represented by ***Q***) in the sentences scoring formula. To begin with, α is set to 0.0 which in effect causes the summarizer to ignore the headline of the document (or ***T*** in the scoring formula) and only considers the document set title (***Q***). The summary generated is shown in Figure A.3. The summary limit is set to 100 words.

```
(1) A window shard, spent shell casings and a bloody mop emerged as
key pieces of evidence Wednesday in the apparent execution-style
murders of the husband and mother of a U.S. district court judge in
Chicago. (2) Returning to her home on the city's north side Monday,
Judge Joan Humphrey Lefkow discovered the bodies of her husband,
Michael Lefkow, 64, and mother, Donna Humphrey, 89. (3) The judge
and her husband briefly received protection last year during the
trial of a white supremacist who later was convicted of trying to
arrange her murder. (4) The judge's husband had recently injured his
leg and had surgery.
```

Figure A.3: Summary generated with α set to 0.0

By examining the sentences in the generated summary and comparing them to the query *Judge Joan Lefkow's Family Murdered*, it can be noted that the name of the judge *Joan Lefkow* appears in sentence 2. Also, the terms *husband* from sentences 3 and 4, and *family* in the query are semantically related. The term *murder* appears explicitly in sentences 1 and 3. When changing the value of α to 0.5 which gives an equal weight to both ***T*** and ***Q***, the summary I get is shown in Figure A.4.

```
(1) A window shard, spent shell casings and a bloody mop emerged as
key pieces of evidence Wednesday in the apparent execution-style
murders of the husband and mother of a U.S. district court judge in
Chicago. (2) Bayless and other officials would not provide specific
details about evidence retrieved from the house, out of concern that
leaks about the investigation might aid the killer or killers. (3)
The judge and her husband briefly received protection last year
during the trial of a white supremacist who later was convicted of
trying to arrange her murder.
```

Figure A.4: Summary generated with α set to 0.5

In the new summary, I have sentences 2 and 4 from the old summary being replaced by sentence 2 in the new summary. Also, it can be noted that an almost equal share of emphasis on terms appearing in both T and Q is applied in the new summary. For example, the word *murder* still appears in sentences 1 and 3. The term *probe* from T is semantically related to the term *investigation* which appears in sentence 2. I try to set the value of α to 1.0. This causes the system to effectively consider only ***T*** and ignores ***Q***. The effect of this is shown in Figure A.5. Note the captured semantic similarity between the term *Security* in sentences 3 and 4 and *Protection* from the document headline.

(1) A window shard, spent shell casings and a bloody mop emerged as key pieces of evidence Wednesday in the apparent execution-style murders of the husband and mother of a U.S. district court judge in Chicago. **(2)** The judge and her husband briefly received protection last year during the trial of a white supremacist who later was convicted of trying to arrange her murder. **(3)** Aspen also asked for a new look at security measures. **(4)** There's real concern that there needs to be a dispassionate, comprehensive look at security arrangements in light of these homicides, Lehman said

Figure A.5: Summary generated with α set to 1.0

Now, by setting the value of α back to 0.0, I shift the focus of the system back to Q. I consider replacing Q with "*risky job*" for the sake of comparison. The effect of this change is shown in Figure A6.

(1) A window shard, spent shell casings and a bloody mop emerged as key pieces of evidence Wednesday in the apparent execution-style murders of the husband and mother of a U.S. district court judge in Chicago. **(2)** In an interview Wednesday for the Chicago Tribune, Lefkow said she always knew her job put her at risk but never thought it would endanger her family. **(3)** Aspen also asked for a new look at security measures. **(4)** There's real concern that there needs to be a dispassionate, comprehensive look at security arrangements in light of these homicides, Lehman said

Figure A.6: Summary generated with α set to 0.0 and Q changed to "*risky job*" for the sake of comparison

By examining the new summary, it appears that the system has captured the similarity between the adjective *risky* and the words *risk* and *endanger* in sentence 2 showing the effectiveness of the measure arTonv_SemSimMeasure implemented in this summarizer.

The same summarizer has also been applied for multi-document summarization. An example showing the effect of this on the document set D0813A from the TAC08 collection is shown in Figure A.7. This summary is generated by setting α to 0.5 which gave the best performance from my own observations.

Reference Summaries

1

In the Nov. 2, 2004 election for Governor of Washington, the first vote count showed Republican Rossi leading by 800 votes. The following day Democrat Gregoire led by 1400; the next day Gregoire by 21,234; and by Nov. 16 Rossi by 19. By Nov 18 all votes were counted and Rossi led by 261. A machine recount gave Rossi a 42-vote edge but a manual recount gave Gregoire a lead of 261. On Dec. 30 Gregoire was declared Governor-elect. On Jan. 7, 2005, with inauguration scheduled for Jan. 12, Rossi filed a lawsuit seeking a new election.

2

Democrat Christine Grigoire was declared winner in the Washington gubernatorial election two months after election day, winning by 129 votes out of 2.9 million cast. Republican opponent Dino Rossi filed a lawsuit seeking the extraordinary remedy of a new election. The initial count after the Nov 2 election had Rossi ahead by 261 votes. The first recount reduced his lead to 42 votes. A second recount put Grigoie ahead. An estimated 60% of Washingtonians vote by mail and absentee ballots can be postmarked as late as election day so counting in close races can drag on for weeks.

3

Until the last days of the campaign, State Attorney General Christine Gregoire (Democrat) was favored to win Washington's 2004 governor's race over wealthy real estate agent Dino Rossi (Republican). Absentee ballots postmarked by Election Day trickled in. Rossi led by 261 votes, triggering a recount required by law if the margin is less than 2000. Following a machine recount, Rossi led by 42 votes. A third count, by hand, took place after Democrats raised sufficient money. Election officials discovered over 700 erroneously rejected or misplaced absentee ballots. Seven weeks after Election Day, Gregoire led by 130 votes, reversing election results.

4

The race for to be governor of Washington state was extremely close. The initial count showed the Republican candidate, Dino Rossi, ahead by 261 votes out of 2.9 million. The first recount, done by machine, showed him ahead by only 42 votes. The second recount, done by hand, gave Democrat Christine Gregoire a 120-vote lead and she was declared the winner on 30 December, fifty eight days after the election. Rossi has filed a lawsuit, alleging that dead people, felons, and other ineligible voters cast ballots and demanding an unprecedented statewide re-vote.The inauguration may be postponed.

System Summary

After a bitter and protracted recount fight in the Washington governor's race, elections officials announced Wednesday that the Democratic candidate, Christine O. Gregoire, was leading her Republican opponent by 10 votes a minuscule margin but a stunning reversal of the Nov. 2 election results. A month and a day after voters went to the polls, the closest governor's race in Washington state history and one of the nation's closest-ever statewide contests lurched forward Friday, as state Democrats announced they had raised enough money to start a third count, this one by hand, of nearly 3 million ballots.

Figure A.7: System summary generated with the WordNet-based summarizer for document set D0813-A from the TAC08 documents collection

Two main methodologies were applied for summarization with Wikipedia. The first uses the term-concepts table while the second employs the strong links features. Both were described and presented in Chapter 5. I present here a sample summary generated by each method. First is shown in Figure A.8 and uses the term-concepts table method. It is applied to document set D1017A from the TAC10 data collection.

Reference Summaries
1 Between 10 and 13 September 1999, a storm named Floyd proceeded from northwest of Barbuda towards Miami, moving at speeds up to 14 mph, with winds progressing from 110 mph to 155 mph, thus upgrading it to a major hurricane. Warnings extended along the Atlantic Coast from South Florida to the barrier islands of South Georgia. Florida Governor Bush declared an emergency, hundreds of thousands, particularly mobile homers, moving inland. Cape Kennedy workers were sent home. FEMA stood ready with ice, potable water, generators, tents, and food, as well as having contractors ready for debris removal and roof repair.
2 In early September, 1999, authorities worried about the growth of Hurricane Floyd as they tracked its movements from the Atlantic toward the U.S. coast. Emergency management officials in Florida began mobilizing and the Governor declared a state of emergency. On September 13, mandatory evacuations were ordered in Brevard County Florida, and coastal Georgia; hundreds of thousands fled their homes. Experts speculated Floyd could become a catastrophic category 5 Hurricane, with winds over 156 miles per hour, the first such storm to hit the coast. Individuals were urged to make homes as hurricane proof as possible.
3 On September 11, tropical storm Floyd became the fourth Atlantic hurricane of the year, moving through the North Atlantic toward the eastern US. Moving away from the Caribbean's Leeward Islands toward Florida, Category 4 Hurricane Floyd, with 131 mph winds that reached 155 mph around the eye, extended 700 miles in all directions. In Florida, emergency management officials mobilized and the governor declared a state of emergency. Mandatory evacuations were ordered in Florida and Georgia, and cruise ships and pleasure craft left for safer waters. FEMA emergency response centers positioned ice, water, cots, tents and emergency food and medical supplies.
4 Floyd became a hurricane on September 10, 1999. It was north of Puerto Rico and the Virgin Islands. By September 13, Floyd had become a Category 4, monster storm, expected to hit the east coast between Miami and Brunswick, Georgia, by September 15. If the winds increase it would be only the third Category 5 storm to hit the U.S. FEMA has opened response centers, formed a liaison team, and pre-positioned supplies in Atlanta. Florida has declared a state of emergency and is mobilizing. Kennedy Space Center is being evacuated. Citizens are urged to prepare and consider voluntary evacuations.

System Summary

(1) The National Hurricane Center in Miami reported that Floyd was expected to intensify today, becoming a Category 3 hurricane, and it could become a Category 4 with sustained winds of at least 131 mph by Tuesday afternoon, when it's expected to reach the northern Bahamas. **(2)** Officials posted warnings for Hurricane Floyd across the central Bahamas and hurricane watches throughout the northwest Bahamas. **(3)** A U.S. hurricane hunter airplane reported Sunday evening that Floyd's winds had topped the 131 mph mark of a very dangerous Category 4 storm. **(4)** Customers streamed into the store to pick up hurricane supplies.

Top Concepts:
Hurricane Floyd, National Hurricane Center, Hurricane Hunters ,Tropical Cyclone, The Bahamas, The United States, Miami, Miles per hour, Fixed-wing aircraft, Flood Alert, Wind, Category 4 Hurricane, Maximum sustained wind, Hunting, Storm, Customer, Tornado Warning, Tornado Watch, Cardinal Direction, Central America

Figure A.8: Summary generated with the Term-concepts table method for document set D1017A from the TAC10 documents collection

The top detected topics within the original document are shown in the above figure. When applying the strong links method to the same document set, I obtain the summary shown in Figure A.9. The noise introduced with the term-concepts table greatly depends on the number chosen for the context size when disambiguating concepts. For this particular example, the top detected concepts appear to correspond better to the reference summaries than in the example provided in section 5.5.7.

(1) Hurricane Floyd got stronger and headed toward the Bahamas Saturday, packing 110 mph winds and leaving weather pundits wondering whether it will hit South Florida this week. **(2)** Five Caribbean islands canceled tropical storm watches Friday night as Floyd, packing winds of 110 mph, moved further out to sea, the National Hurricane Center in Miami reported. **(3)** The National Hurricane Center in Miami reported that Floyd was expected to intensify today, becoming a Category 3 hurricane, and it could become a Category 4 with sustained winds of at least 131 mph by Tuesday afternoon, when it's expected to reach the northern Bahamas.

Top Concepts:
Hurricane Floyd, National Hurricane Center, Hurricane Hunters ,Tropical Cyclone, Tropics, Tropical Cyclone Warning and Watches, Flood Alert, Storm, Confederate states of America, Saffir-Simpson Hurricane Scale, Wind, Weather, Out to Sea, South Florida Metropolitan Area, The Bahamas, Miami

Figure A.9: Summary generated with the strong links method for document set D1017A from the TAC10 documents collection

APPENDIX B

Penn Treebank Tags and Stanford Typed Dependencies

Provided here is a list showing the Treebank tags that were used during the parsing of sentences in Chapter 6 and Chapter 8. The list is followed by the Stafornd Typed Dependencies which were used in Chapter 6 for describing how the processes of sentences splitting are performed.

Tag	**Description**
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential *there*
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
NP	Noun Phrase
PDT	Predeterminer
PP	Prepositional Phrase
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
S	simple declarative clause
SBAR	Clause introduced by a (possibly empty) subordinating conjunction

SYM	Symbol
TO	*to*
UH	Interjection
VP	Verb Phrase
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WHNP	Wh-noun Phrase
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Stanford Typed Dependencies[16]

rcmod*: *relative clause modifier	
"I am the man you love"	rcmod(man, love)
dobj: direct object	
"She gave me a raise"	dobj(gave, raise)
ccomp: clausal component	
"He says that you like to swim"	ccomp(says, like)
dep: dependent	
"Then, as if to show that he did it … "	dep(show, if)
conj: conjunct	
"Bill is big and honest"	conj(big, honest)
Nsubj: nominal subject	
"The baby is cute"	nsubj(baby, cute)

[16] All definitions and examples are taken from Stanford Dependencies Manual http://nlp.stanford.edu/software/dependencies_manual.pdf

APPENDIX C

Publications

The following publications have been published while working on this thesis. They are added here to provide a reference to the different sections covered in this thesis and the experiments that were conducted. They also give details about any ideas that may have been included briefly within this thesis. A brief description about how each paper relates to the work in this thesis was provided in Chapter 1 section 1.2.

Sentences Simplification for Automatic Summarization

Abdullah Bawakid and Mourad Oussalah
School of Engineering
Department of Electronic, Electrical and Computer Engineering
University of Birmingham
{ axb517 , M.oussalah }@bham.ac.uk

***Abstract*— In this paper, we emphasize the need for conserving space within sentences by introducing a Sentences Simplification Module (SSM). The module is aimed to shorten the length of sentences via either splitting or compression. We describe how the module is integrated in a Wikipedia-based summarization framework. We highlight the performance differences obtained from introducing such a module by running a series of evaluations.**

Keywords; Summarization; Sentences Simplification; SSM; links analysis ; strong links

I. Introduction

The goal of summarization systems is to provide summaries containing as much information as possible meeting the user's needs within a confined space determined by the previously-set summary limit. Since most of the recently developed summarization systems are extractive, they are inherently limited to only the sentences that exist in the original test documents. A sentence may contain important information in part and non-relevant information in another. Also, a sentence that may be central to the theme of a document may contain a mixture of new and redundant data to what is already available in the summary. It is therefore necessary to enforce a methodology that allows for conserving more space within sentences and including as much content as possible without sacrificing novelty or imposing redundancy. For this, it is probably best to view how linguists and discourse analysts perform their studies on text and sentences in particular. Their analysis usually begins by separating clauses and phrases within a sentence to identify their features and properties [1]. In this paper, we describe how a similar step was introduced by integrating a module for automatic sentences simplification into the Wikipedia-based summarization framework we previously developed for the purpose of better information extraction.

The rest of this paper is organized as follows. In section 2 we briefly discuss the related work in the field. In section 3 we give an overview on the built SSM and its main components. In section 4 we describe the summarization methodology we adopt and how SSM is integrated in the process. Section 5 discuss the evaluation performed on the summarization system and compare its performance against other baselines. In section 6 we summarize and conclude our work.

II. Related Work

Sentences Simplification has been applied in the literature to different applications with different motivations. In [2] and [3], it has been used to form sentences which are easier to read for humans and new language learners in particular. Their target was to create sentences which are grammatically correct, short and cohesive. Sentence simplification has also been applied in summarization systems to shorten the length of sentences. The CLASSY summarization system described in [4], [5] and [6] for example employs a sentence compression module by applying a set of rules to all sentences before choosing the summary candidate sentences. Cut-and-Paste in [7] and other summarizers in [8], [9] and [10] employ a similar approach. In [11], the authors applied a sentences compression module after choosing the candidate sentences in a post-processing step.

In the mentioned summarization systems, compression is applied independently as a separate process to all sentences and is not affected by how candidates are selected. It is possible that two sentences containing an important piece of information are compressed by removing that important content from both. It would be optimal to remove the redundant info from only one sentence while keeping it in the other if both sentences were to be included in a summary. Implementing this would require a system having the capability of applying a dynamic set of rules to different sentences based on what is already contained in the summary.

It was suggested in a pilot study in [12] that summarization systems implementing compressions would have an edge over those that do not if the compression took into account the different references and relationships among sentences. In [13], an approach was applied to tackle the mentioned issue. Their system would apply a set of rules for trimming sentences and creating multiple versions of each sentence using the rules described in [14]. After creating multiple compressed versions of each sentence, the core summarizer would consider all versions of each sentence as potential summary candidates. To choose the optimal candidate, the system would check the redundancy of the sentence against the current summary and select the least redundant. Another system with a similar approach was implemented and described in [15]. Both systems emphasized preserving only important content and do not necessarily factor in the semantic relationships between the documents content. Redundancy checking is based on BOW methods and do not capture the semantics similarity and relatedness between the different sentences and the concepts they carry. In our system, we also employ a similar methodology by producing multiple simplified versions of sentences. However, the focus here is rather on simplifying sentences while in the same time preserving as much relevant semantic content as possible. For that, the Sentences Simplification Module (SSM) we developed is embedded in our Wikipedia-based summarizer and an iterative process is added for handling redundancy.

III. Overview of SSM

Before simplifying a sentence, it is necessary first to devise some means for interpreting its text. The interpretation can be syntactic, lexical or semantic. Focus here is on syntactical interpretation of sentences. For that, dependency tree of the sentence is drawn with help from Stanford's Parser[1] which adopts the Penn Treebank conventions. With the tree drawn, one can apply any set of rules to make the changes desired to the sentence. In effect, the rules may allow for simplifying a long and complicated sentence through compression or splitting into several sentences. The simplified sentences, along with the original, become summary candidates and the choice is based on the contained semantic information in each and the redundancy and relevancy to what is already available in the current state of the summary. Factors like sentences length (before and after the split/compression) and the existence of dominant concepts/words/phrases play a role in making decisions dynamically. In the next two subsections we describe the two main processes involved in SSM, namely Sentences Splitting and Sentences Compression.

A. Sentences Splitting

Sentences with different syntactic formats would need to be handled differently with different rules applied to achieve the splits desired. Based on the generated parse tree and grammatical relations for the original sentence and the boundary terms found in a sentence, it is decided what rule to apply. The boundary terms are chosen to be *who*, *which*, *that*, *and* and *or*. Many of the rules mentioned here have been applied in the literature in different studies for different systems including [2], [16] and [17]. The following scenarios show the Tregex [18] patterns[2] detected and processed by SSM for sentences splitting with an example provided for each.

Scenario 1

Sentence: *The man who ate the poisoned food died yesterday.*
Parse: *(ROOT (S (NP (NP (DT The) (NN man)) (SBAR (WHNP (WP who)) (S (VP (VBD ate) (NP (DT the) (VBN poisoned) (NN food)))))) (VP (VBD died) (NP (NN yesterday)))))*

Tregex Pattern: *SBAR , NP & < (WHNP << who) & << VP*

Used Modifiers: *rcmod(man-2, ate-4)*

Sentences after Split: *The man died yesterday. The man ate the poisoned food.*

[1] http://nlp.stanford.edu/software/lex-parser.shtml
[2] Syntactic patterns are written in this paper in the tree searching language *Tregex* which encodes the different tree relations

In the above example, the boundary term *who* appears before the phrase "*ate the food yesterday*". When viewing the phrase structure tree for the sentence, a phrase of type *SBAR* which has *WHNP* as one of its descendents is detected. This indicates that a potential split is in place. The split is achieved by separating the *SBAR* clause from the rest of the sentence resulting in two sentences, with one being incomplete. These two sentences are:

The man died yesterday.

who ate the poisoned food.

To complete the second sentence, we look at the phrase proceeding the boundary term *who*, which is of type *VP*. With the relation *rcmod(man-2, ate-4)*, one can tell that man is the word that should be preceding the main verb in the incomplete sentence.

Scenario 2

Sentence: *I have read the books which you bought last week.*
Parse: *(ROOT (S (NP (PRP I)) (VP (VBP have) (VP (VBN read) (NP (NP (DT the) (NNS books)) (SBAR (WHNP (WDT which)) (S (NP (PRP you)) (VP (VBD bought) (NP (JJ last) (NN week))))))))))*
Tregex Pattern: *SBAR , NP & < (WHNP << which) & << (VP , NP < VBD)*
Used Modifiers: *rcmod(books-5, bought-8)*
Sentences after Split: *I have read the books. You bought the books last week.*

In this sentence, the boundary term *which* is proceeded by the Noun Phrase (*NP*) *you*. For this scenario, we find the following *VBD*, which is *bought*, and use the relation *rcmod* to find the subject that is being referred to by that verb. We obtain *books* as the answer. The *NP* of *books* is used to complete the second sentence.

Scenario 3

Sentence: *I have read the books that you bought last week.*
Parse: *(ROOT (S (NP (PRP I)) (VP (VBP have) (VP (VBN read) (NP (DT the) (NNS books)) (SBAR (IN that) (S (NP (PRP you)) (VP (VBD bought) (NP (JJ last) (NN week)))))))))*
Tregex Pattern: *SBAR , NP & < (IN << that) & << (VP , NP < VBD)*
Used Modifiers: *dobj(read-3, books-5), ccomp(read-3, bought-8)*
Sentences after Split: *I have read the books. You bought the books last week.*

The boundary term in this example is *that*. It is found underneath an *SBAR* and has an *NP* following it. The *VBD bought* is used with the relation *ccomp* to find its complement: *read*. The *dobj* relation is then employed to specify the direct object of the verb which is *books* in the given example.

Scenario 4

Sentence: *I have read the books you bought last week*
Parse: (ROOT (S (S (NP (PRP I)) (VP (VBP have) (VP (VBN read) (NP (DT the) (NNS books))))) (S (NP (PRP you)) (VP (VBD bought) (NP (JJ last) (NN week))))))
Tregex Pattern: S < (NP . VP) , S < (NP.VP)
Used Modifiers: dep(read-3, bought-7), dobj(read-3, books-5)

Sentences after Split: *I have read the books. You bought the books last week.*

This example does not contain a boundary term. Two *S* clauses appear with *NP* and *VP* for each. The segmentation takes place by separating the clauses from each other. Since the verb *bought* is dependent on *read*, we use the relation *dobj* to determine the *NP* following *bought*.

Scenario 5

Sentence: *The team won the golden medal and achieved the highest team score of the season.*
Parse: *(ROOT (S (NP (DT The) (NN team)) (VP (VP (VBD won) (NP (DT the) (JJ golden) (NN medal))) (CC and) (VP (VBD achieved) (NP (NP (DT the) (JJS highest) (NN team) (NN score)) (PP (IN of) (NP (DT the) (NN season))))))))*
Tregex Pattern: *VP , NP < (VP < VBD $ VP) < (CC < and | < or)*
Used Modifiers: *conj_and(won-3, achieved-8)*
nsubj(won-3, team-2)
Sentences after Split: *The team won the golden medal. The team achieved the highest team score of the season.*

The boundary term in the above example is *and* which separates two *VP*s. The relation *conj_and* is used to link the main verb of the second clause with the first and thus find the shared object.

Scenario 6

Sentence: *The infected rabid fox eventually dies, but a simple scratch can spread the virus to other animals or people.*
Parse: *(ROOT (FRAG (S (S (NP (DT The) (JJ infected) (JJ rabid) (NN fox)) (ADVP (RB eventually)) (VP (VBZ dies))) (, ,) (CC but) (S (NP (DT a) (JJ simple) (NN scratch)) (VP (MD can) (VP (VB spread) (NP (DT the) (NN virus)) (PP (TO to) (NP (JJ other) (NNS animals) (CC or) (NNS people)))))))))*
Tregex Pattern: *S < (NP . VP) $ (CC < but) $ S < (NP.VP)*
Used Modifiers: conj_but(dies-6, spread-13)
Sentences after Split: The infected rabid fox eventually dies. A simple scratch can spread the virus to other animals or people.

The boundary term in the above example is *but*. It is located between two clauses of type *S*. Both clauses have *NP*s followed by *VP*s as their children. This pattern triggers a possible split which is achieved with help from the relation *conj_but* to construct the new sentences.

B. Sentences Compression

We adopt the Trimmer algorithm described in [14] and [19] by applying some of its syntactic compression rules. The original Trimmer algorithm aims to transform sentences into headline-style phrases by detecting patterns in a sentence parse tree and removing certain nodes from the tree based on the applied rule and detected pattern. The goal here is to generate sentences, not headlines, which are short, cohesive and grammatically correct. Therefore, we implement only some of the rules that would be most suitable for producing complete sentences and discard those resulting in headline-styled text. The rules being used are the following with an example provided for each:

Keep the leftmost S root and remove its siblings

Keep the leftmost *S* Root which has both *NP* and *VP* in the sentence and the remove the rest.
Sentence: The Libyan leader and his wife were in good health, *Mossa Ibrahim told a press conference*.
Parse: *(ROOT (S (S (NP (NP (DT The) (JJ Libyan) (NN leader)) (CC and) (NP (PRP$ his) (NN wife))) (VP (VBD were) (PP (IN in) (NP (JJ good) (NN health))))) (, ,) (NP (NNP Mossa) (NNP Ibrahim)) (VP (VBD told) (NP (DT a) (NN press) (NN conference))) (. .)))*
After compression: The Libyan leader and his wife were in good health.

Remove Time Expressions

Remove temporal expressions from sentences. This is achieved by deleting the *PP* node which has an *NP* child with a *Time* word as one of its leaves. The deletion takes place by removing the Prepositional Phrase (*PP*) with all of its children.
Sentence: After the raid took place *on Saturday around 8:00 pm*, Ibrahim took a group of journalists to the site of the house.
Parse: *(ROOT (S (SBAR (IN After) (S (NP (DT the) (NN raid)) (VP (VBD took) (NP (NN place)) (PP (IN on) (NP (NNP Saturday))) (PP (IN around) (NP (CD 8:00) (NN pm)))))) (, ,) (NP (NNP Ibrahim)) (VP (VBD took) (NP (NP (DT a) (NN group)) (PP (IN of) (NP (NNS journalists)))) (PP (TO to) (NP (NP (DT the) (NN site)) (PP (IN of) (NP (DT the) (NN house)))))) (. .)))*
After Compression: After the raid took place, Ibrahim took a group of journalists to the site of the house.

Remove Conjunctions

For any sentence containing *and* or *but* as a CC, we remove the preceding phrase of *but* and proceeding for *and*.
Sentence: NATO continued its precision strikes against Gaddafi regime military installations in Tripoli overnight *but would not confirm the Libyan claim about the assassination attempt.*
Parse: *(ROOT (S (NP (NNP NATO)) (VP (VP (VBD continued) (NP (PRP$ its) (NN precision) (NNS strikes)) (PP (IN against) (NP (NNP Gaddafi) (NN regime) (JJ military) (NNS installations))) (PP (IN in) (NP (NNP Tripoli))) (ADVP (RB overnight))) (CC but) (VP (MD would) (RB not) (VP (VB confirm) (NP (DT the) (JJ Libyan) (NN claim)) (PP (IN about) (NP (DT the) (NN assassination) (NN attempt))))))))*
After Compression: NATO continued its precision strikes against Gaddafi regime military installations in Tripoli overnight.

Remove Complements

We remove *IN* nodes which have the term *that* in their leaves.
Sentence: The alliance acknowledged *that* it had struck a command and control building
Parse: *(ROOT (S (NP (DT The) (NN alliance)) (VP (VBD acknowledged) (SBAR (IN that) (S (NP (PRP it)) (VP (VBD had) (VP (VBN struck) (NP (DT a) (NN command) (CC and) (NN control) (NN building)))))))))*
After Compression: The alliance acknowledged it had struck a command and control building

XP over XP

XP here refers to either *NP* or *VP*. For first *NP*-over-*NP* or *VP*-over-*VP* where inner *XP* is the first and leftmost child, we keep the left child and remove all of the child siblings. Note that the *XP* child must be the first and leftmost child of the parent *XP* for the rule to apply.

Sentence: A woman *whose husband killed himself with a circular saw in Plymouth earlier this week* was bludgeoned to death.
Parse: *(ROOT (S (NP (NP (DT A) (NN woman)) (SBAR (WHNP (WP$ whose) (NN husband)) (S (VP (VBD killed) (NP (PRP himself)) (PP (IN with) (NP (NP (DT a) (JJ circular) (NN saw)) (PP (IN in) (NP (NNP Plymouth))))) (NP (RBR earlier) (DT this) (NN week)))))) (VP (VBD was) (VP (VBN bludgeoned) (PP (TO to) (NP (NN death)))))))*
After Compression: A woman was bludgeoned to death

Remove PP under SBAR

PP expressions appearing under *SBAR*s are removed.
Sentence: The administration said it has deployed several countermeasures to reduce oil dependence *such as supporting research in alternative energy sources*.
Parse:
After Compression: The administration said it has deployed several countermeasures to reduce oil dependence.

Remove SBAR

We remove *SBAR*s in this step as illustrated in the following example.
Sentence: NATO forces *whose air strikes could not stop Gaddafi attacks on civilians* decided to supply rebels with weapons.
Parse: *(ROOT (S (NP (NP (NNP NATO) (NNS forces)) (SBAR (WHNP (WP$ whose) (NN air)) (S (NP (NNS strikes)) (VP (MD could) (RB not) (VP (VB stop) (NP (NNP Gaddafi) (NNS attacks)) (PP (IN on) (NP (NNS civilians)))))))) (VP (VBD decided) (S (VP (TO to) (VP (VB supply) (NP (NNS rebels)) (PP (IN with) (NP (NNS weapons))))))) (. .)))*
After Compression: NATO forces decided to supply rebels with weapons.

Remove PP

We remove all *PP* nodes in the tree with this rule.
Sentence: NATO continued its precision strikes *against Gaddafi regime military installations in Tripoli overnight*.
Parse: *(ROOT (S (NP (NNP NATO)) (VP (VBD continued) (NP (PRP$ its) (NN precision) (NNS strikes)) (PP (IN against) (NP (NNP Gaddafi) (NN regime) (JJ military) (NNS installations))) (PP (IN in) (NP (NNP Tripoli) (JJ overnight)))) (. .)))*
After Compression: NATO continued its precision strikes.

At first, the rules mentioned above are applied to a sentence independently. And then in a second iteration, they are all applied to the sentence in sequence with the output of one rule being fed to the next. The aim of both operations (applying rules independently and in order) is to produce as many compressed and valid versions of a sentence as possible.

IV. Summarization Methodology

The basic summarization methodology applied here relies on a modified version of the Wikipedia-assisted summarizer that utilizes the strong weighted links approach described in [20]. The major change is in the introduction of SSM and an iterative process handling redundancy. Figure 1 shows the architecture of the updated summarizer with the iterative process inside the box. After preprocessing the documents, clusters of sentences are formed with each cluster having the original sentence in addition to its simplified versions as generated from SSM. Afterwards, all sentences are given a score using the weighted links method. Then, an iterative process is applied after which a summary is produced.

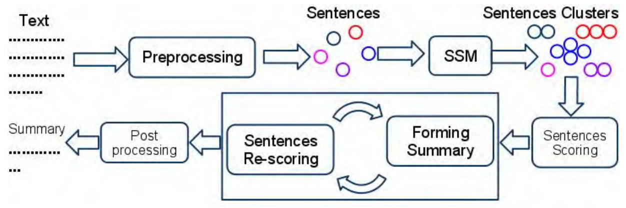


Figure 1. Architecture of the SSM-based Summarizer

The sentences scoring stage relies on other process to complete, namely: identifying the concepts within sentences, measuring the relatedness between concepts and sentences, and identifying the features used for scoring sentences. In the next subsections we outline each of these processes.

A. *Identifying the Concepts*

We use an exact match measure where explicitly mentioned Wikipedia concepts within each sentence are detected. A concept having multiple spellings and synonyms should still be detected by the system as a single concept. This is due to the integration of redirect links within the thesaurus and the mapping algorithm that associates sentences with the concepts they contain.

B. *Measuring the Relatedness between Concepts*

For every explicitly detected concept, it is possible to devise a vector of related Wikipedia articles through the strong links method. The vector would contain the related articles and the weight assigned to each based on the detected link types between them. We compute the relatedness between any two concepts using the cosine measure formula as follows:

$$rel(a,b) = \frac{\sum_{i,j} a_i \times b_j}{\sqrt{\sum_{i=1}^{n}(a_i)^2} \times \sqrt{\sum_{j}^{m}(b_j)^2}} \quad (1)$$

where a and b are the two concepts to be compared and a_i and b_j are the weights associated with their related articles as extracted from Wikipedia using the Strong Links method.

C. *Measuring the Relatedness between Sentences*

Each sentence would have a vector of the concepts detected in it using the exact match method. The semantic relatedness between two sentences is computed by the following formula:

$$Srel(Sent_1, Sent_2) = \frac{rel(A,B)}{PairsCounter} \quad (2)$$

Where *Sent*$_1$ and *Sent*$_2$ refer to Sentence1 and Sentence2 respectively, *A* is the concepts set in Sentence1, *B* is the concepts set in Sentence2, and *PairsCounter* is the number of concepts pairs compared.

D. Feature Selection

Each sentence is tagged with several features. These features are used to compute a score determining the sentence importance. The main features used are the following:

Overlap with the Topic/Query: We consider the overlap between each sentence and the topic of its documents set. We take into account concepts overlap when assigning a score to each sentence. Synonyms and concepts with alternative spellings are considered as a single concept in our system with the help of the Wikipedia thesaurus and the custom matcher.

Concepts Dominance: When computing a score for each sentence based on this feature, we consider how pertinent the sentence concepts to the important concepts with the document set.

Sentence Position: Sentences appearing in the top 20% and the bottom 20% portion of a document are given position scores 50% larger than the others.

E. Sentences Scoring

Each sentence is assigned a score representing its importance. The score for each sentence (i), is simply the linear combination of the weights given for each feature. The formula used for assigning a score to each sentence is:

$$\text{Score(i)}=\frac{(\alpha Srel(s_i, \text{T}) + \beta Srel(s_i, Q))\ n(s_i)\ P(s_i)}{N} \quad (3)$$

Where:

- N = the total number of sentences
- $\alpha + \beta = 1$
- $n(s_i)$ = The number of sentences that have semantic relatedness score bigger than a pre-defined threshold value
- $P(s)$ = either 1 for sentences appearing at the top and end of the document, or 0.5 for the rest.
- $Srel(s_i\ ,T)$ and $Srel(s_i\ ,Q)$ are for the Semantic Relatedness between the Title and the Query, respectively, and the sentence (i)

The rationale behind the preceding is similar to what was proposed for the WordNet-based summarizer which was described in [21]. The main difference here is that the system deals with Wikipedia concepts, and the ones detected within a sentence either directly through EM or indirectly are taken into account when scoring the sentence.

F. Integration of SSM

The approach we use here assumes that an optimal summary would contain the largest amount of the most useful and relevant concepts within a limited space. This is implemented in the system through the introduction of an iterative process enforcing this idea. In the original summarizers we previously built, each sentence was given a score signifying its importance based on a set of features: overlap with topic/query, concepts dominance and sentence position. With the iterative process employed here, two of the mentioned features become dynamic, namely overlap with topic/query and concepts dominance. After each iteration, the top ranking sentence is added to the summary and its concepts are identified. The identified concepts are then removed from the source documents and all remaining sentences are rescored. The iterative process can be summarized by the following steps:

1- After scoring all sentences for the first time, we obtain a ranked list of candidate sentences with the top being with the highest score.

2- We remove the top highest scoring sentence from the Candidate Sentences List (CSL) and add it to the summary. Only one sentence should exist in the summary at this stage.

3- The cluster of the sentence that was just included in the summary is added to a Sentences Exclusion List (SEL). The cluster should contain the non-simplified version of the sentence in addition to all of its simplified versions.

4- We detect all the concepts present in the sentence that was just added to the summary and add them to a Concepts Exclusion List (CEL).

5- We re-score all remaining sentences taking two factors into account: First being sentences in SEL should be ignored. Second is any occurrence of a concept that exists in CEL should be ignored too.

6- Add the highest scoring sentence to the summary and verify the summary length does not exceed the given limit. If it does not, go to step 3. Otherwise go to the post-processing stage and produce the summary.

Note that in step 5, redundancy is implicitly enforced by counting concepts only once and preferring sentences with a high density of concepts. Simplified sentences that are short and contain important and relevant concepts would still be selected as the approach ensures that no concept repetition within the summary takes place.

V. Evaluation

To evaluate the implemented system, we used the TAC10 dataset with the same parameters as those used in the system we participated with in TAC10 [22]. Two other systems were used as baselines. The first utilizes the Trimmer algorithm and creates a single compressed version of each candidate sentence. The summary is then formed by aggregating the compressed versions of the highest ranking sentences. The second baseline is the summarizer implemented for TAC10 using the strong weighted links without SSM. The results obtained with the ROUGE metric are illustrated in Table 2 and Figure 2.

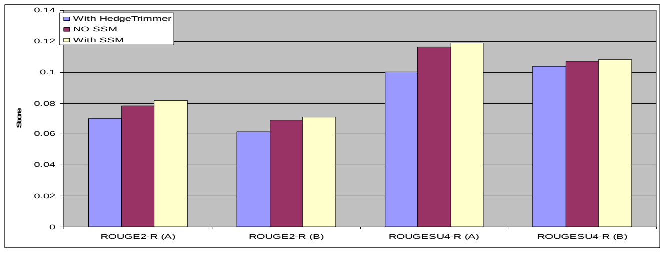


Figure 2: Comparison of the ROUGE results obtained for the different systems

TABLE 2: THE ROUGE RESULTS OBTAINED SUGGEST PERFORMANCE IMPROVEMENT WITH THE SSM-BASED SUMMARIZER.

Evaluation	Trimmer	NO SSM	With SSM
ROUGE2-R (A)	0.07011	0.07883	**0.08173**
ROUGE2-R (B)	0.06160	0.06901	**0.07101**
ROUGESU4-R (A)	0.10026	0.11889	**0.11917**
ROUGESU4-R (B)	0.10401	0.10702	**0.10815**

It can be noted from the results that the introduction of the SSM-based system led to various levels of improvements to the ROUGE results when compared against the original Wikipedia-based summarizer. This goes along with the intuition that compressing sentences should increase the capacity of a summary. With the increased capacity, it is vital to have a dynamic features selection that can aid with sentences selection. This is evident by examining the results of the Trimmer baseline where compressing all sentences in the summary as a post-processing stage caused a loss to the system's performance.

VI. CONCLUSION

In this paper, we briefly described a module we implemented for simplifying sentences and producing multiple splits and compressed versions of each sentence. The simplification module, SSM, is aimed to help with summarization by segmenting sentences to remove non important parts while retaining relevant parts for inclusion in the summary. For this purpose, the syntactical interpretation of sentences allows for patterns detection and applying a set of rules to simplify sentences whenever possible. After obtaining multiple simplified versions of each sentence, another module within the Wikipedia-based summarizer chooses the most important and least redundant sentence to include in the summary. An evaluation was performed and the obtained and reported results indicate an achieved improvement in the summarizer.

REFERENCES

[1] J. P. Gee, *An Introduction to Discourse Analysis: Theory and Method*, 1st ed. Routledge, 1999.

[2] A. Siddharthan, "Syntactic Simplification and Text Cohesion," *Research on Language and Computation*, vol. 4, no. 1, pp. 77-109, Mar. 2006.

[3] J. Carroll, G. Minnen, Y. Canning, S. Devlin, and J. Tait, "Practical Simplification of English Newspaper Text to Assist Aphasic Readers," in *In Proc. of AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, 1998, pp. 7-10.

[4] J. Conroy and S. Judith, "CLASSY Query-Based Multi-Document Summarization," in *Proceedings of the 2005 Document Understanding Conference (DUC-2005) at NLT/EMNLP 2005*, Vancouver, Canada, 2005.

[5] J. Conroy, J. Schlesinger, D. Schlesinger, and J. Goldstein, "Back to Basics: CLASSY 2006," in *Proceedings of the Sixth Document Understanding Conference (DUC)*, 2006.

[6] J. Conroy, J. Schlesinger, and D. Schlesinger, "CLASSY 2007 at DUC 2007," in *Proceedings of the Seventh Document Understanding Conference (DUC)*, Rochester, New York, 2007.

[7] H. Jing and K. Mckeown, "Cut and paste based text summarization." 2000.

[8] S. Blair-Goldensohn et al., "Columbia University at DUC 2004," 2004.

[9] H. Jing, "Sentence Reduction for Automatic Text Summarization," *IN PROCEEDINGS OF THE 6TH APPLIED NATURAL LANGUAGE PROCESSING CONFERENCE*, p. 310--315, 2000.

[10] H. Liu, Q. Zhao, Y. Xiong, L. Li, and C. Yuan, "The CIST Summarization Systems at TAC 2010," in *Proceedings of the Text Analysis Conference (TAC)*, Gaithersburg, Maryland USA, 2010.

[11] H. Daume III and D. Marcu, "Bayesian Multi-Document Summarization at MSE," *Workshop on Multilingual Summarization Evaluation (MSE)*, 2005.

[12] C.-Y. Lin, "Improving summarization performance by sentence compression: a pilot study," in *Proceedings of the sixth international workshop on Information retrieval with Asian languages - Volume 11*, Stroudsburg, PA, USA, 2003, pp. 1–8.

[13] D. Zajic, B. Dorr, J. Lin, and C. Monz, "A sentence-trimming approach to multidocument summarization," *IN PROC. OF DUC*, p. 151--158, 2005.

[14] B. Dorr, D. Zajic, and R. Schwartz, "Hedge Trimmer: A Parse-and-Trim Approach to Headline Generation," 2003.

[15] L. Vanderwende, H. Suzuki, C. Brockett, and A. Nenkova, "Abstract Beyond SumBasic: Task-Focused Summarization with Sentence Simplification and Lexical Expansion," *Information Processing and Management*, vol. 43, no. 6, pp. 1606-1618, 2007.

[16] E. Ong, J. Damay, G. Lojico, K. Lu, and D. Tarantan, "Simplifying Text in Medical Literature," *Journal of Research in Science, Computing and Engineering*, vol. 4, no. 1, 2008.

[17] S. Jonnalagadda and G. Gonzalez, "Sentence Simplification Aids Protein-Protein Interaction Extraction," *1001.4273*, Jan. 2010.

[18] R. Levy and G. Andrew, "Tregex and Tsurgeon: tools for querying and manipulating tree data structures," *IN LREC 2006*, 2006.

[19] D. Z. Bonnie and B. Dorr, "BBN/UMD at DUC-2004: Topiary," *IN PROCEEDINGS OF THE 2004 DOCUMENT UNDERSTANDING CONFERENCE (DUC 2004) AT NLT/NAACL 2004*, p. 112--119, 2004.

[20] A. Bawakid and M. Oussalah, "Using features extracted from Wikipedia for the task of Word Sense Disambiguation," in *2010 IEEE 9th International Conference on Cybernetic Intelligent Systems (CIS)*, 2010, pp. 1-6.

[21] A. Bawakid and M. Oussalah, "A Semantic Summarization System: University of Birmingham at TAC 2008," in *Proceedings of the First Text Analysis Conference*, 2008.

[22] A. Bawakid and M. Oussalah, "Summarizing with Wikipedia," in *Proceedings of the Text Analysis Conference (TAC)*, Gaithersburg, Maryland USA, 2010.

Summarizing with Wikipedia

Mourad Oussalah, Abdullah Bawakid

School of Engineering
Department of Electronic, Electrical and Computer Engineering
University of Birmingham
{M.oussalah , axb517 }@bham.ac.uk

Abstract

This paper describes a query-based multi-document summarizer that was built to participate in the update summarization task of TAC10. The system relies on a thesaurus extracted from Wikipedia and uses it as its underlying ontology. The concepts which are detected within the documents are used as weighted features to score the document sentences. The relationships previously defined in the thesaurus between the different concepts help in finding the most important concepts within a document or a set of documents. Sentences are ranked based on the scores they have been assigned and the summary is formed from the highest ranking sentences till the 100-word limit is reached. The evaluation results and the performance of the system are described. The total number of the submitted runs by all participants is 43.

Keywords: Semantic Similarity, Semantic Relatedness Wikipedia, Text Summarization, Information Retrieval

1. Introduction

The Text Analysis Conference (TAC) is one of the well-known workshops in the field of Natural Language Processing which provides the infrastructure necessary to evaluate different methodologies with different tasks. In TAC10, we participated in the Guided Summarization task with two different runs. The aim of the task is to provide short summaries for a set of newswire articles. The generated summaries are not to exceed 100 words each. This year's task is different from last year in that the participants are asked to a deeper semantic analysis of the source documents instead of simply relying documents words frequencies to select the important concepts. For this, a list of categories and important aspects for each category are given and it is asked that the summary provided should cover all of the mentioned aspects if possible in addition to any other information related to the topic.

The "update" part of the task is similar to that of TAC09 and TAC08. For a given set of documents, the participants are asked to write two summaries, one for set A and another for set B. A topic statement is provided in addition to the Categories aspects which have been added to the task only this year. The participants are asked to write 100-word summary for set A using the given topic statement and the specified category. For set B, a 100-word update summary is to be generated assuming that the user has already read the set of articles in set A.

To enhance the representation of the documents to summarize in each set, the developed system described in this paper applies a set of rules to expand the document representation with the help of an external ontology. In our participation in the Summarization task of TAC08, we relied on WordNet as an external ontology [1]. In this year, we used Wikipedia instead. Wikipedia has several advantages over what WordNet has to offer. The coverage and breadth of Wikipedia is larger than that of WordNet. In addition, it is more up-to-date. Using the concepts extracted from Wikipedia is especially useful with short topic statements provided in the task for each set. Also, they are used to detect the most dominant concepts within a document and the inter-connection between these dominant concepts within a document set and the given topic.

In our system, we use the Wikipedia ontology to build a thesaurus containing a list of Wikipedia's concepts. To determine the relationship between all of the extracted concepts from Wikipedia, we used the internal links, categories structure and other rules. These concepts are used to aid in extracting the dominant concepts within each document and documents set, and the association strength of the extracted concepts. The ontology we used along with a description of how it was built was reported in our earlier work[2].

The rest of this paper is structured as follows: an overview of the related work followed by a description of how the concepts ontology was built and extracted from Wikipedia. Then, we describe our system and how it was applied to this year's task. Next, we present the evaluation results and discusses the rank, the strength and the limitations of our system. Finally, the paper is concluded with a potential future work.

2. System Overview

The system developed for the summarization task is extractive. Each sentence is assigned a score signifying its importance based on its extracted features. The summary is then generated for sets A by ranking the sentences based on their assigned scores in a descending order and choosing the top n sentences till the maximum word-limit is reached. The stages involved for creating summaries are summarized in the following subsections:

2.1 Preprocessing

The first stage in the framework is to preprocess all fed documents by cleaning them and then parsing them to extract the text and topics and then tokenizing the terms and splitting the sentences. The stop words are then removed.

2.2 Identifying the Concepts

Two methods have been utilized to detect concepts by employing the built Wikipedia-thesaurus and its extracted features. First one is through an exact match measure where explicitly mentioned concepts within each sentence are detected. A concept having multiple spellings for a concept and synonyms should still be detected by the system as a single concept. This is due to the integration of redirect links within the thesaurus and the mapping algorithm that associates sentences with the concepts they contain. As for ambiguous terms and concepts, the system implements the Weighted Strong Links method that was described in [2].
In the second method we examine each term within a sentence and replace it with its concepts vector is through the term-concepts table. The concepts vector has a weight associated to each concept signifying its relatedness with the term. After generating a concepts vector for each term, we group all concepts vectors within a sentence by summing the scores of the individual concepts that are repeated. This in effect applies word sense disambiguation as relevant concepts are boosted and given a higher score in the merged concepts vector. For example, the concept "Fox" has two meanings: "Fox (Animal)" and "Fox (Broadcasting company)". Similarly, the concept "Dog" is associated with "Mammals". In the sentence "A fox attacked a dog", the meaning "Fox (Animal)" is boosted.

2.3 Features Selection

Each sentence is tagged with several features. These features are used to compute a score determining the sentence importance.

Overlap with the Topic: In our system, we consider the overlap between each sentence and the topic of its document set. We take into account both the concepts overlap and the terms overlap when assigning a score to each sentence. Synonyms and concepts with alternative spellings are considered as a single concept in our system with the help of the Wikipedia thesaurus and the custom matcher.

Concepts Dominance: The explicitly mentioned concepts within a document set which are most frequent and the topic concepts are considered to be the most important. When computing a score for each sentence based on this feature, we consider how pertinent the sentence concepts to the important concepts with the document set. We use the relevancy degrees between the concepts which are precomputed in the Wikipedia thesaurus for achieving this task.

Sentence Position: The system assumes that sentences appearing at the top and bottom of a document have more chances of being important than the rest. Therefore, sentences appearing in the top 20% and the bottom 20% portion of a document are given position scores 50% larger than the others.

2.1 Measuring the Relatedness and Similarity between Sentences

Each sentence would have a vector of the concepts detected in it using the exact match method. In addition, it would have another vector of concepts generated from merging its individual terms concepts as extracted from the term-concepts table. When evaluating two sentences, we consider both vectors to compute the similarity and relatedness between them. The semantic relatedness is computed by the following formula:

$$Srel(Sent1, Sent2) = \frac{rel(A,B)}{PairsCounter}$$

Where Sent1 and Sent2 refer to Sentence1 and Sentence2 respectively, A is the concepts set in Sentence1, B is the concepts set in Sentence2, and PairsCounter is the number of concepts pairs compared. This formula can be applied to both vectors individually.

3.5 Summary Generation

Knowing what features to use in the system, it is possible to assign a score for each feature in each sentence. A sentence score comprises of its Topics scores, the relevancy of these Topics with the dominant ones, the overlap between the sentence and the rest of the sentences in a document, and the position of a sentence in the document. After scoring all sentences, the summary is formed by ranking the sentences in a descending order based on their scores, and adding the sentences one by one to the summary till the 100-word limit is reached.

After adding the last sentence to the summary and reaching the mentioned word limit, the sentences are re-ordered according to their appearance in the original documents they were taken from. The last sentence in the summary is then truncated to enforce the 100-word limit. At last, we applied a custom set of rules we developed to remove non-important data from some sentences such as date stamps and writers references appearing at the beginning of some sentences.

3. Evaluations

The provided dataset for the update task is composed of 46 topics divided into five categories. Each topic has a title, category, and 20 relevant documents divided equally into two sets: A and B. Documents in set A precede chronologically those in set B. Participants are asked to submit a summary for each set. They are also given the option of submitting up to two runs for each team.

We participated with two runs. The ids of our runs are 14 and 19. The term-concepts table method was used with run 19 while strong links method was used for 14. In Table 1, the ranks obtained by the system in the different evaluation methods are displayed. The total number of runs the system is compared with is 43.

	Manual	ROUGE-2	ROUGE-SU4	BE
Run 14	11	13	14	10
Run 19	13	17	15	15

Table 1: Evaluation results for the Update Task showing ranks of the two submitted runs 14 and 19 relative to the 43 submitted runs

4. Conclusion

In this paper, we briefly described the methodology that was implemented in our system for this year's Update task. We outlined how Wikipedia was used, the features that we focused on, and how the summaries were constructed. The results obtained show that the performance of our system is competitive when compared with the other teams systems, although there is still room for improvement. Creating a redundancy/diversity matcher and finding a better method to set their thresholds, and implementing better measures to utilize the found concepts and better understand what they refer to through a deeper linguistic analysis than what is performed here are potential future work we intend to focus on.

References:

[1] A. Bawakid and M. Oussalah, "A Semantic Summarization System: University of Birmingham at TAC 2008," in *Proceedings of the First Text Analysis Conference (TAC 2008)*, 2008.

[2] A. Bawakid and M. Oussalah, "Centroid-based Classification Enhanced with Wikipedia," in *The Ninth International Conference on Machine Learning and Applications 2010*, 2010.

[3] L. Qiu, M. Kan, and T. Chua, "A Public Reference Implementation of the RAP Anaphora Resolution Algorithm," *cs/0406031*, Jun. 2004.

[4] A. Bawakid and M. Oussalah, "Using Features Extracted from Wikipedia for the Task of Word Sense Disambiguation," in *9th Conference on Cybernetic Intelligent Systems*, 2010.

Using Features Extracted from Wikipedia for the Task of Word Sense Disambiguation

Abdullah Bawakid and Mourad Oussalah
School of Engineering
Department of Electronic, Electrical and Computer Engineering
University of Birmingham
{ axb517 , M.oussalah }@bham.ac.uk

Abstract—**In this paper, a method using features extracted from Wikipedia for the task of Word Sense Disambiguation (WSD) is presented and evaluated. A term-concepts table constructed from Wikipedia and the redirect links is described. With its help, the Wikipedia internal links along with the categories structure are used to compute the relatedness between any two concepts through a two-level process: a term-concepts expansion followed by a links-based expansion. The result is a ranked list of concepts which are most related to the ambiguous term given the context it exists in. For the evaluation experiment, the benchmark is constructed from a segment of the internal links of Wikipedia. The evaluation results obtained suggest that introducing links analysis and the categories structure to the built term-concepts table provide improvement to the accuracy of the method in the WSD task.**

Keywords; Word Sense Disambiguation; WSD; Wikipedia; links analysis ; Categories ; strong links

I. Introduction

In English, and many other languages, a word may carry more than one meaning. For example, the term *Fall* may refer to the autumn season of the year, the movement caused by the earth gravity, or it can be the academic term occurring usually around the autumn season. The correct meaning of a word is usually determined based on the context it is placed in. In previous work, it has been hypothesized that the correct sense of a polysemous word can be determined by the surrounding words accompanying it [1],[2]. We follow here the same hypothesis in our system by using the surrounding words to help decide the right sense of the target word.

Our approach relies on the use of the vast and highly organized human knowledge existing within Wikipedia giving it a major advantage over other approaches using smaller thesauruses such as WordNet or Open Directory Project (ODP). Due to its openness and structure, Wikipedia is not suitable for being used directly as is by machines and its content needs to be analyzed first with semantic processing tools. In our semantics-extraction framework, we treat each Wikipedia article as a unique Concept and use its title as a label. The content of the article is used to help build a relationship vector between the article terms and its title. The formed term-concept vector along with the links and Categories structure existing within Wikipedia are analyzed in a separate step to aid in computing the relatedness score between any terms, or even text fragments. In this work, we highlight an optimized version of our method being used in the application of Word Sense Disambiguation (WSD).

The rest of this paper is organized as follows. In section 2 we briefly describe how the term-concepts table is constructed and the boosting algorithm we apply to it afterwards. In section 3 we give an overview on the link types we focus on, and the filtering we apply to the categories and links. In section 4 we describe our WSD system and the stages involved in it. Section 5 gives an overview on the related work. In section 6 we conclude our work and some potential areas to investigate in the future.

II. Construction of the Term-Concepts Table

The preprocessed version of Wikipedia is analyzed to leverage the articles contents, their titles, redirect links and the categories structure. First, all stop-words are removed from the articles content and stemming using Porter's stemmer algorithm is applied. The remainder terms are then used to serve the purpose of representing all of Wikipedia Concepts. This was achieved by examining the terms distributions within each article and computing the weight for each word in the form of TFIDF which is one of the most common weighting methods used to describe documents in the vector space model. TFIDF factors two aspects for each term: its frequency within each document (represented as TF) where the higher the TF in a document the more chance that it is important within that document. The second factor is the Inverse Document Frequency IDF where a word is deemed more important in a document if it doesn't appear in many of the test collection documents giving it a higher IDF value.

A. Term-Concepts Table

In essence, we map all the terms existing within each article to all the Wikipedia Concepts creating a vector for each term whose elements are the l2-normalized term weights within each Concept. These weights resemble how much the terms contribute to each concept they are attached to. We rank the concepts each term belongs to based on the formed weights in a decreasing order to form the Term-Concept table. The top concepts in the list are the most relevant ones to the term. For example, the term Birmingham has the following associated concepts *history of Birmingham*, *Birmingham (the English City)*, *Birmingham Alabama (the American City)*, *Arts in Birmingham*, *Timeline of Birmingham History*, *Birmingham City University*, *Barry Vincent Jackson*, *B Postcode Area*, *Birmingham Local Elections*, *Economy of Birmingham*, *Birmingham Business Journal*, etc. One can notice that the covered range of different concepts varies from the city name (in UK and USA), to events that occurred in one of the two cities, to a person name who owned a theatre the English Birmingham.

After building the table, we perform a two-level update to it through a series of iterations focusing on boosting the weight scores for some terms based on their appearances within the titles and redirects links. This is explained further in the following section.

B. Concepts Boosting

In the *Birmingham* term example, one can also notice the occurrence of the term *Birmingham* in many of the Concepts titles or the text of the articles they belong to. This is not always the case for some of the other generated concepts. Take the word *Unhappy* as an example. Some of the concepts the word is related to do not have any occurrence of the word *Unhappy* in their titles or even the article text they represent. For example, the concept *Depression (mood)* does not have any reference to the word *Unhappy* in its text or its title, yet it is still related to it. With the sole help of the Term-Concept table previously built, the concept *Depression (mood)* would not appear in the list of related concepts for the term *Unhappy* because the TF for that term in the concept's article text is zero. In a similar way, some concepts titles may contain the keyword *Unhappy* in their titles which should give them a higher tendency to be more related to the term than many other concepts. For instance, the concepts *Unhappy Consciousness*, *Unhappy Triad* and *Unhappy Happiness* are assumed to be more related to the term *Unhappy* due to them all sharing the key term *Unhappy*.

To tackle the above-mentioned issues, we thereby apply a two-level Boosting process as a following step after generating the concepts vector using the term-concept table for a given term. In essence, we make use of the large number of Redirect links existing within the Wikipedia structure by analyzing the keywords existing within the title of each Redirect link in addition to the titles of the articles they link to. In the first boosting level for a term or a group of terms ***w***, we hypothesize that a redirect link ***r*** containing only ***w*** in its title should link to a concept ***c*** that is highly related to ***w*** regardless of whether the concept ***c*** has ***w*** in its title or text. In other words, ***w*** and ***c*** should have a high relatedness score which is achieved through the boosting performed in our algorithm. This is done by assigning a score to ***c*** based on the value of a variable we call *FirstLevelBoost*. In a similar way, we apply the same idea to the concepts titles ***ct*** and ***w*** to generate a relatedness score for ***ct*** using the same variable *FirstLevelBoost*.

In the second boosting level, we examine the occurrence of the term or group of terms ***w*** in the concept title ***ct*** or the redirect link ***r*** that points to it. If ***ct*** or ***r*** contains ***w*** in addition to some other terms, we increment the relatedness score of ***c*** by a value correlated with *SLB*. The resulted relatedness score from the second level boost will always be less than the first level boost. Also, as the number of terms appearing within the title or redirect link increases, the amount of boost being applied inversely decreases. This is reflected in the following formula being applied to generate a value for the *SLB*.

$$SLB_t = ((SLB - 1)^{SecondLevelBoostAdj} + 1) \qquad (1)$$

$$SecondLevelBoostAdj = | c_t | / | t_c | \qquad (2)$$

In (2), it is displayed how SecondLevelBoostAdj is computed. In it, we have $| \boldsymbol{c_t} |$ as the number of terms that exist within the concept title (or the redirect link), and $| \boldsymbol{t_c} |$ as the number of times the term ***tc*** appears within the concept title.

III. Wikipedia Links and Categories Structure

Wikipedia contains a large amount of structured data. It has structured pages for ambiguous terms listing their possible meanings with links to the articles describing them. It also has structured categories attached to each article. The categories have parents and/or children relationships defined among them. Articles belonging to the same category have generally similar outline and structure. In addition, over 86 million links exist within it linking articles with each other. These links are of different types and can be a representative for some form of relationship between articles with each other. The categories too with their hierarchy can help better enhance the definition of the semantic relationship between the articles.

In this work, we focus on using the links and categories structure in Wikipedia to enhance the features we previously extracted. Wikipedia contains different types of links. There are interlanguage links linking to a version of the article in different language. There are internal links linking to other pages within the same Wikipedia language. There are interwiki links pointing to other pages within the Wikimedia project but not necessarily to Wikipedia articles. There are also external links to pages outside the wikimedia project. Many other types of links exist too within the articles such as section links, date links, and template links. Our focus here is on the internal hyperlinks in the articles text pointing to different English articles in Wikipedia.

Not all the internal links are of the same significance. Some links may be more reflective of the relatedness of an article to another than many others. To illustrate this, take the two links *Basketball court* and *Peripheral Vision* existing within the article about the famous *Basketball* sport game as an example. Intuitively, the former link is more related to the article than the second. It is thus important to apply some form of filtering to the article links to reduce the resident noise and embrace those that link to most related articles. Therefore, we devise our own links-filtering module. The module's goal is twofold. First, it reduces the number of noisy or unimportant links by focusing only on high-valued links. Second it enhances the overall efficiency of the system since the total number of links to be evaluated and analyzed will be reduced. This is especially evident for those articles that contain a large number of incoming links such as the article about the famous company *Google* which has over 70,000 incoming links. The analysis of such large number of links for all the articles in Wikipedia would require a large amount of computing resources and time and is not simply efficient.

In the filtering module, we classify the internal links into several levels signifying their importance based on our own observations. As for the categories, we attempted to utilize the category structure within Wikipedia directly on its own but realized that even though some categories are narrow and indicate strong relatedness between their articles, some other categories are broad and not as useful as many others. For example, the category *Historiography* is broad and has 129 pages. Among these pages are *Silver Age* and *Source Text* which can not be said to be strongly related to each other. Due to the generality of some categories and because we still think the categorization can still be useful especially for some narrative categories, we chose to filter the categories we use with the internal links. Figure 1 shows the links types we defined sorted based on the weights they carry in a decreasing order. In general, it possible to divide the defined link types into three categories: Mutual Links where two articles directly link to each other, One Link with shared a Parent Category, and See Also links which are usually appended to most of the articles in Wikipedia. We next define and describe these link types along with the weight level we assigned to each:

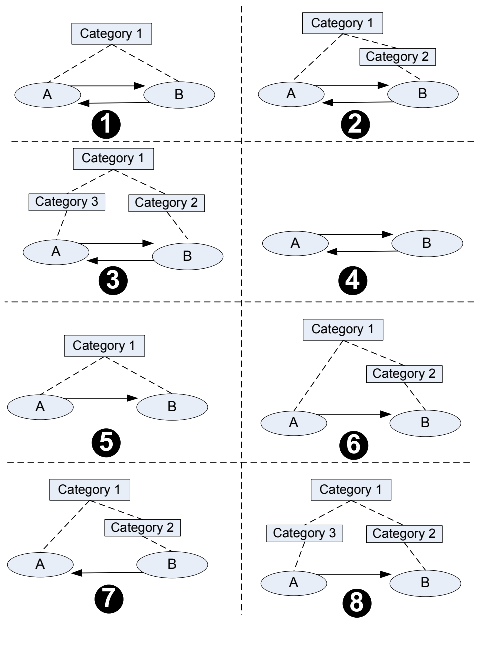


Figure 1. Link Types defined sorted based on their weights in descending order

A. Mutual Linking

When article A contains a link or more pointing directly to article B, and article B contains a link or more pointing to article A, we consider these links for the two articles to be of a high value signifying a strong relatedness between the two articles. An example for this is the Basketball and Slam dunk articles which both contain links pointing to each other and they are closely related. If the two articles share one or more parent categories, they are expected to be much more related than if they were not. Thus, we classified the mutual link types into four classes 1-4 as shown in Figure 1. In 1, both articles directly share a parent category. In 2, the parent category of one article is a subcategory of the parent category of the second article. In 3, both articles share a grandparent category that is exactly one category-level away from the articles. In other words, the articles belong to at least two categories whose parents are the same. In 4, no shared parent or grandparent category is found and thus only the reciprocal links are considered. All of the four link types 1-4 resemble strong relatedness between the two articles in each case when compared with the types 5-8 but of varying weight. The weights we assigned to each link type are illustrated in Table 1.

B. Sharing Parent-Category with One Link

As highlighted above, merely having a shared category between two articles can indicate a strong relevancy between the articles. However, this can not be applied as a general rule due to the breadth and generality of some categories. Therefore, we adopted the "at least" one link sharing rule as a filtering mechanism. We also expanded it to include grandparent categories in some cases, namely 6, 7 and 8 in Figure 1. As a general rule for the link types covered in this category of links, we say that when Article A points to Article B or B links to A AND both articles belong to the same category (or grandparent category), we have a potential strong relevancy between the two articles A and B. An example for this category of links is the articles titled *Great Depression* and *Panic of 1893* which both belong to the category *Financial Crisis* and the former article has a link pointing to the second. Both articles discuss the economic depressions that occurred before the Second World War. However, Until the *Great Depression*, *Panic of 1893* was considered the harshest in the history of the United States. The relevancy between these two articles is thus greater than the relevancy between either and the rest that only share one parent category with either of them such as the articles *Bad Bank* and *Bank Run* both under *Financial Crisis* category.

Table 1: Weights assigned for the different links types

w_1	3	w_2	2.75	w_3	2.5	w_4	2.25	w_5	1.75
w_6	1.5	w_7	1.45	w_8	1.25	w_9	3.75	w_{10}	3.25

C. "See Also" Links

These links are usually added manually by the Wikipedia volunteer editors to the end of the articles in Wikipedia and refer the readers to other semantically related topics to the current article. We give a high weight to these types of links and label them with w_9 in Table 1. We also give a high weight to the inverse of the See Also links labeling them with w_{10} in Table 1.

IV. The Process of WSD

The task of WSD is to automatically predict the right sense for a specific term in the given context. In our system, we use the local features presented in the given context and the previously-extracted Wikipedia features to achieve this task. Figure 2 gives an overview of the modular design in our WSD system. In each instance run, a text document is fed to the system which can be a text fragment, a sentence, a paragraph or a whole document. In the text document, a single term is marked as the target word to be disambiguated, and is displayed as a separate input in Figure 2 for illustration. The provided document is then processed sequentially in each module as a pipeline and the final output of the system is the target sense for the marked word. In essence, the system predicts the correct concept (or sense) for the target term by applying the term-concepts vector to the target term and then scoring each concept according to our analysis of the strong links. The result is a ranked list of scored concepts with the top concept in the list being the most likely sense for the target term and is produced as the final result. In the following subsections, each module in the WSD is described.

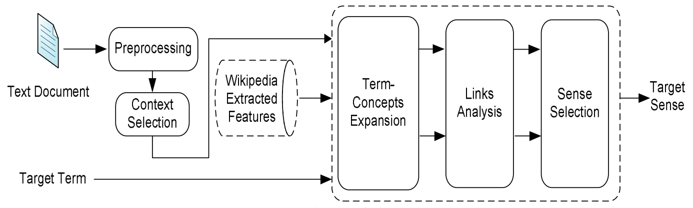


Figure 2: an Overview of the WSD process

A. Preprocessing and Context Selection

Based on the format of the input text document, its content is parsed first and its terms are extracted. Stop words are then removed from the document. This is followed by a preprocessing step in which the marked target term is highlighted to determine its context based on its surrounding words. If the supplied text document is in the form of a short text fragment (less than a predefined number of words), all of its surrounding terms are considered. Otherwise, we consider extracting 2n words surrounding the target term, n words before and n words after, which we call Context Terms (CT) in the following steps. Then, stemming is applied on all of the context terms.

B. Term-Concepts Expansion

After obtaining CT from the Context Selection module, the term-concept vector is applied on the resulting CT and also the target term. The number of concept lists that would be generated is |CT| + 1. If we label each concept list with $C_{i=1\ldots|CT|}$, we would have the concepts list defined as:

$$C_i = \{c_{ij}\}_{j=\{1\ldots V\},\ i=1\ldots|CT|} \quad (3)$$

Where i is the number identifying the concept list, j is the concept number in the list and V is the total number of concepts in the concept list C_i. As for the target word concepts list TW, we define it as:

$$TW = \{g_k\}_{k=\{1\ldots M\}} \quad (4)$$

Where g_k is the concept numbered k in the target term concepts list and M is the number of concepts in the list.

C. Links Analysis and Sense Selection

When analyzing the internal links present in an article, we consider only those that fall into the category of one of the above-mentioned types. We use the links as part of a process to compute the relatedness between two articles A and B. This is achieved by two methods: (i) directly examining the links present in both A and B and computing a score for each link. The second is (ii) forming two sets of articles SA and SB where each set would contain the most relevant articles to A and B respectively and then computing the similarity between the two sets using the Cosine distance. The following subsections describe how both methods fit in our process and highlight the advantages and disadvantages of both.

Simple Links Analysis

In this method, we analyze the links present in the article content of each concept in C_i against all concepts in TW. We use the previously extracted Wikipedia features for this task, in particular the articles links and the categories structure. Based on the weight assigned for each link type, we assign each concept c_{ij} a score cw_{ij} presenting its importance. This transforms the formed concept list C_i into another expanded list C_i' that has a score associated with each of its members.

$$C_i' = C_{ik} = \{ (c_{ij},c_{wij})_k \}_{j=\{1\ldots V\},\ i=1\ldots|CT|,\ k=\{1\ldots M\}} \quad (5)$$

Where k denotes the corresponding concept g_k that the pair (c_{ij},cw_{ij}) points to. In other words, for each concept c_{ij}, we compute its links-strength score against all the concepts g_k and associate the score wc_{ij} to each comparison. Therefore, the total number of entries in C_i' should be |TW| * |Ci|.

After expanding the concepts list C_i to C_{ik} and producing a score cw_{ij} for each concept c_{ij}, the next step is to score all concepts in TW based on the values generated in C_{ik}. This is achieved by expanding TW into a new weighted list we call TW' which has a score gw_k associated to each concept g_k.

$$TW' = \{ (g_k, gw_k) \}_{k=\{1\ldots M\}} \quad (6)$$

We compute the score for each concept g_k by summing the generated scores in C_i for all the related concepts to g_k. This can be translated to:

$$gw_k = \Sigma_{j=\{1..V\}} \Sigma_{i=\{1..|CT|\}} (cw_{ijk}) \quad (7)$$

Essentially, all the concepts in TW' will have varying scores which are used to determine how related the concept to the context domain chosen early in the process. The final chosen concept g_k (or sense) will have the maximum score.

$$\max_k (gw_k) \quad (8)$$

Links-Based Expansion

In the previous method, we consider the cases where two related articles have a link being directly shared between them. While this may be true in many cases, there are still some cases where two related articles do not directly share a link and they are still semantically related. Therefore, we propose performing the comparison between any two articles indirectly through expanding the two articles into two lists and then comparing the two lists with each other. This can be translated for our WSD process into the following:

$$eTW = \{ Gk \}\ k=\{1\ldots M\} \quad (9)$$

eTW is the expanded list for TW. It contains a list G_k of related articles for each possible sense g_k. The list of G_k is formed as follows:

$$G_k = \{ \ (gc_w , f(g_k , gc_w)) \}_{w=\{1...V\}} , f(gk,gcw) >0 \quad (10)$$

In (10), gc_w is any article that is related to g_k. The function $f(g_k,gc_w)$ measures the relatedness between the two concepts g_k and gc_w based on the link analysis and the weight assigned to each link. V refers to the total number of available concepts for the set G_k after the expansion. We apply the same process to all the concepts existing within C_i and then group them all together in one set summing the score for repeated concepts if they occur in more than one set. We thus obtain the following as a result of the later expansion of C_i:

$$eC_i=\{(rc_{ijw} , f(c_{ii} , \ rc_{ijw}))\}_{j=\{1...V\}, i=1...|CT|, w=1...Q}, f(c_{ii} , rc_{ijw}) >0 \quad (11)$$

Where w refers to the number of the related concept rc_{ijw}, rc_{ijw} is the concept related to c_{ij}, and Q is the total number of related concepts. The resulted score $f(c_{ii} , \ rc_{ijw})$ can be written as tw_{ijw} for abbreviation. We sum all of the resulting related concepts eC_i into one list eC where repeated concepts rc_{ijw} are grouped into one by summing their weight:

$$eC = \{ (rc_v , tw_v) \}_{v=1...|D|} \quad (12)$$

$$tw_v= \Sigma_{w=\{1..Q\}} (tw_{ijw}), \text{ where } rc_w=rw_v \quad (13)$$

Where rc_v is a unique concept in eC_i, D is the total number of *unique* concepts in eC_j, tw_v is the weight given (after computing the sum) for concept rc_v.

After obtaining the list eC of related articles derived from the context terms, and a list G_k for each meaning of the target term, we compute the distance between eC and each Gk using the cosine distance measure. The final answer for the WSD problem would be the concept carrying the number k where G_k is most similar to eC.

$$\max_k (\text{dist}(G_k, eC)) \quad (14)$$

V. Evaluation

The datasets being used for evaluating WSD systems greatly depend on the variations of the parameters for the systems being evaluated. Therefore, we created our own benchmark which is most similar to those devised in the recent work of Turdakov [3] as well as others [4] [5]. We use the manually created links in Wikipedia as the basis for our dataset. The internal links of Wikipedia look in the following form: [[Part1 | Part2]] where Part1 is the text in the hyperlink or the title of the page being linked to while Part2 is the text displayed to the reader while reading the article containing this link. An example for this is the text fragment “With colors such as shades of [[Brown (color) | brown]]” in the article titled *Rabbits*. Clicking the word brown in that fragment redirects the reader to the article *Brown (color)*.

We therefore chose the mentioned technique to construct a dataset of 1000 examples of ambiguous terms along with their correct meanings and the paragraphs that contain them from Wikipedia. During the evaluations, we chose 20 words surrounding the target word TW as the context terms. We performed three experiments on the built dataset to test our method and the effect of introducing the chosen links: (i) the simple Links analysis method, (ii) the links-based expansion and (iii) centroid-based method that does not rely on links. For the third method, we simply expand each term to generate a list of corresponding concepts using the term-concepts table. Then, we compute the centroid for the lists generated from the context terms along with the target term. The centroid is used to create a single list of ranked concepts based on their scores. The predicted sense would be the one with the highest score in the generated list.

TABLE 2: THE ACCURACY OF ALL IMPLEMENTED METHODS INCLUDING (I) SIMPLE-LINK ANALYSIS, (II) LINKS-BASED EXPANSION, AND (III) CENTROID BASED METHODS.

	Top	**Top-2**	**Top-3**
(I)	64.82	75.4	81.02
(II)	**75.31**	**87.89**	**91.52**
(III)	63.75	75.3	82.50

The best result we obtained was with the link-based expansion method (75.31). This shows that including the strong links for analysis does indeed improve the overall accuracy of the system especially when compared with the centroid-based method (63.75). The accuracy obtained for all the methods during the evaluations is shown in Table 2. Since we produce a ranked list of weighted potential meanings in each method, we considered computing the chances of having the right sense in the top-2 and 3 senses of the produced list. The results we obtained show that the accuracy of the best method increases to 91.52 when considering the top-3 senses in the list. Furthermore, the centroid-based method produces a better result than the simple link-based analysis when considering the top-3 senses.

The testing during the evaluation was strict in that we only count a correctly identified sense if it were the one labeled in the constructed corpus. In a small number of cases, the system correctly predicted the right sense for some terms or gave an alternative answer to the one provided by the Wikipedia volunteer who edited the link. Even in that case we considered the found link as a mismatch when computing the results since it didn’t match the exact meaning referred to by the link. An example for this is the link in the sentence “WSVN and KTTV are [[Fox Entertainment Group |Fox]] affiliates”. The system chose Fox Broadcasting Company instead, which is what the Wikipedia editor should have chosen for the mentioned link.

It is possible that the results obtained above can be improved by expanding the number of context terms considered during evaluation. Also, the number of related documents that were considered after the links-based expansion was 20 at maximum. Increasing that number or finding a better model that gives a better compromise between efficiency and accuracy would be optimal and is expected to produce even better results.

VI. Related Work

Different types of WSD systems have been proposed in the past. Some systems are knowledge-based relying on external knowledge sources and dictionary definitions. Lesk [6] for example used dictionary definitions of terms to compare with the surroundings of the target word and decide the right sense for the target word. Navigli et al [7] used WordNet and other lexical resources to form structural sense specifications for each word in a context and selected the best hypothesis based on a set of rules they defined in their method. Reference [1] used a WordNet-based relatedness measure to compute the semantic relatedness between the context words of an ambiguous term and all the possible senses for that term. The sense giving the highest relatedness score is then chosen as the correct sense.

Some other systems are data-driven relying on statistical probabilities computed from a given sense-annotated corpus. For instance, Gliozzo et al [8] exploited kernel methods to model sense distinctions in a supervised method. In [9], related non ambiguous words were used for the constructing of examples retrieved from the web. These examples were then processed to replace the non ambiguous words with ambiguous ones giving example contexts for the different senses of the ambiguous words. [10] relied on the idea that ambiguous words can have different translations in other languages. They used such collections of parallel text in their method to annotate the different senses of ambiguous words.

The method we propose here has several advantages over the mentioned above. First, using knowledge database or a dictionary in a system will render the system limited by the breadth, coverage and accuracy of the knowledge they depend on. We used Wikipedia which is the largest known encyclopedia to mankind. The size of Wikipedia is increasing and so its coverage, breadth and accuracy making it most suitable for the job than others. Second, our method here is monolingual and no parallel data is required. After preprocessing the Wikipedia dump for the first time, we use the extracted features in our system for as many processes as needed. The accuracy obtained by our system as shown in the evaluation results is comparable, or exceeding, to those of the systems mentioned above.

With respect to Wikipedia, other systems investigated the use of its links and category structure. Mihalca [2] used a supervised method employing the internal links of Wikipedia for the purpose of building a corpus with annotated senses in a WSD task. After building the sense-annotated corpus, they matched the senses to the definitions in WordNet. In [4], the authors proposed a method that extracts a feature vector for each of the occurrence of an ambiguous word from the links in Wikipedia. We used a similar method to build our evaluation corpus for this work.

In [5], a method was investigated using one type of internal links in Wikipedia: mutual links. In [3], several link types were proposed and weighted. We extend these two methods in our work by including more important link types and using the category structure within Wikipedia to filter the chosen links. We also employ a test-concept table that provides a more comprehensive list of related articles for any given word. This allows the consideration for the senses which are not explicitly displayed in the Disambiguation pages of Wikipedia. We show in the results that even with the breadth introduced from applying the test-concept table to the terms, the method still obtains good evaluation results.

The term-concepts table we use in our method is similar in principle to the concepts matrix built in [11]. However, we apply our boosting algorithm to consider the alternative names mentioned in Wikipedia's redirect links. We also integrated the structure of the categories as well as the classified and weighted internal links in Wikipedia to our method. One of methods we evaluate is based on a centroid formed from the built term-concepts table for the sake of comparison.

VII. Conclusion and Future Work

We have presented a method for WSD utilizing the extracted features from Wikipedia, namely: the term-concepts table, categories structure and strong links. We have classified the strong links and gave a weight to each type. We then evaluated the method and presented the results we obtained. The good results we have encourage us to apply the same methodologies to other applications including text classification or categorization. Also, it is possible to use an optimized version of our method on other open knowledge sources such as Wiktionary. We plan to do that next and compare the obtained results with the Wikipedia-based method.

References

[1] S. Patwardhan, S. Banerjee, and T. Pedersen, "UMND1: unsupervised word sense disambiguation using contextual semantic relatedness," *Proceedings of the 4th International Workshop on Semantic Evaluations*, Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 390-393.

[2] R. Mihalcea, "Using Wikipedia for Automatic Word Sense Disambiguation," *North American Chapter of the Association for Computational Linguistics (NAACL 2007)*, 2007.

[3] D. Turdakov and P. Velikhov, "Semantic Relatedness Metric for Wikipedia Concepts Based on Link Analysis and its Application to Word Sense Disambiguation," *SYRCoDIS*, CEUR-WS.org, 2008.

[4] R. Mihalcea and A. Csomai, "Wikify!: linking documents to encyclopedic knowledge," *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, Lisbon, Portugal: ACM, 2007, pp. 242, 233.

[1] A. Fogarolli, "Word Sense Disambiguation Based on Wikipedia Link Structure," *Semantic Computing, 2009. ICSC '09. IEEE International Conference on*, 2009, pp. 77-82.

[6] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone," *SIGDOC '86: Proceedings of the 5th annual international conference on Systems documentation*, Toronto, Ontario, Canada: ACM, 1986, pp. 26, 24.

[7] R. Navigli and P. Velardi, "Structural semantic interconnections: a knowledge-based approach to word sense disambiguation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, 2005, pp. 1075-1086.

[8] A. Gliozzo, C. Giuliano, and C. Strapparava, "Domain kernels for word sense disambiguation," *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Ann Arbor, Michigan: Association for Computational Linguistics, 2005, pp. 403-410.

[9] X. Wang and D. Martinez, "Word sense disambiguation using automatically translated sense examples," *Proceedings of the International Workshop on Cross-Language Knowledge Induction*, Trento, Italy: Association for Computational Linguistics, 2006, pp. 45-52.

[10] M. Diab, "Relieving the data acquisition bottleneck in word sense disambiguation," *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, Barcelona, Spain: Association for Computational Linguistics, 2004, p. 303.

[11] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using Wikipedia-based explicit semantic analysis," *IN PROCEEDINGS OF THE 20TH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, 2007, pp. 1606--1611.

Centroid-based Classification Enhanced with Wikipedia

Abdullah Bawakid and Mourad Oussalah
School of Engineering
Department of Electronic, Electrical and Computer Engineering
University of Birmingham
{ axb517 , M.oussalah }@bham.ac.uk

Abstract— **Most of the traditional text classification methods employ Bag of Words (BOW) approaches relying on the words frequencies existing within the training corpus and the testing documents. Recently, studies have examined using external knowledge to enrich the text representation of documents. Some have focused on using WordNet which suffers from different limitations including the available number of words, synsets and coverage. Other studies used different aspects of Wikipedia instead. Depending on the features being selected and evaluated and the external knowledge being used, a balance between recall, precision, noise reduction and information loss has to be applied. In this paper, we propose a new Centroid-based classification approach relying on Wikipedia to enrich the representation of documents through the use of Wikpedia's concepts, categories structure, links, and articles text. We extract candidate concepts for each class with the help of Wikipedia and merge them with important features derived directly from the text documents. Different variations of the system were evaluated and the results show improvements in the performance of the system.**

Keywords-component; Classification, Semantics, Wikipedia, Categorization, text enrichment (key words)

I. Introduction

The amount of newly created information in electronic form increases in a large pace everyday. In particular, the available text on the web created the necessity to implement different methodologies to organize information into different useful forms. Among the various approaches that have been designed for managing the information is Automatic Text Classification (ATC) which is the process of assigning previously defined classes to documents. ATC has been used recently in many web applications including search engines queries [1] and web documents classification [2]. These applications usually require fast training and classification in addition to high precision and recall.

Many traditional text classification systems focus on Bag of Words (BOW) techniques which represent the documents or classes with weighted features extracted from documents terms and their frequencies. Among the most popular BOW techniques are SVM [3], Neural Networks [4], knn [5], Naïve-Based [6] and centroid-based methods [7],[8]. Centroid-based methods were generally found in the literature to be faster and more efficient than most of the rest of the methods. Their precision and recall were also lacking when compared with other methods such as SVN [8]. In a recently published study, a new centroid-based method was proposed and found after evaluation to give better accuracy than many of the other state-of-art BOW approaches [7].

While the efficiency and performance of many BOW methods may be somewhat high for tasks when the category of a group of documents can be identified with a few distinct keywords appearing in the belonging documents, this is not always the case. For example, consider the case when a class labeled "Abnormal Psychology" has several training documents in which none has the keyword "Hyperthymesia" (meaning superior memory). If a document discussing Hyperthymesia were to be classified, a BOW method would not be able to distinguish the relationship between the class "Abnormal Psychology" and the word "Hyperthymesia". On the other hand, a human with good background knowledge in Hyperthymesia should be able to tell which class the document actually belongs to. Also, consider the case when two consecutive words such as "Cat Fish" provide a new meaning different from the two separate words. With only traditional BOW methods, multi-word concepts are usually misinterpreted or simply omitted. Hence, the use of external knowledge to enrich classification methods should help address similar scenarios where semantic understanding of the content of the documents and the relationship between its contents and the different classes is needed. This semantic analysis is especially important when the training or testing documents are short in length providing not much enough info for training with BOW methods.

In this paper, we describe a novel system that employs Wikipedia as its underlying knowledge base in a unique way. The large number of concepts and diverse domains covered in Wikipedia makes it most suitable for the task. Instead of mapping the documents text to a concept or a small group of concepts as done in most of the previous work, we map it to all of the previously-processed Wikipedia concepts. This is achieved by first processing all Wikipedia articles and extracting the relationship between each of its terms and all the concepts existing within Wikipedia. In essence, this forms a term-concepts table. Then, we extract the categories structure within Wikipedia and analyze its links. Furthermore, we employ a centroid-based method directly on the documents contents and give the terms weights based on inter-class, inner-document and inter-document features. The result of all the mentioned steps (Concepts, Categories and Text) is then combined to form prototype vectors for each class during the training stage. The classification uses the formed vectors to decide which class each Test Document (TD) belongs to in an efficient way. Our experimental results on the 20-newsgroups dataset and the ODP collection demonstrate that our classifier performs very well when compared to previous methods.

The rest of the paper is organized as follows: section 2 discusses the related work. Section 3 elaborates the extracted features and the design of our method. Section 4 describes how the extracted features are used in the different implemented heuristics. Section 5 presents the evaluation results and discusses our findings. Finally, in section 6 we conclude this paper with possible research direction in the future.

II. RELATED WORK

An increasing amount of work has been recently applied to enriching text representations for different applications including classification, clustering, information retrieval and clusters labeling. Different kinds of knowledge bases have been used too for the different applications. In [9], WordNet was used to enhance the Classification of text documents by improving the Rocchio algorithm. Their method was supervised and required manual annotations of terms vectors. In [10], WordNet was used for the task of documents clustering. They used WordNet synsets to enrich the representation of documents but without word sense disambiguation. The results they obtained did not show improvements with the use of synsets.

Wikipedia was also used in different applications. In [11], Wikipedia was used to build a thesaurus for use in specific domains. The focus was on using Wikipedia's internal and redirects links for this task. with small emphasis on the rich relations and hierarchy available in Wikipedia. Also in [12] and [13], a method was proposed and evaluated that uses Wikipedia and ODP to obtain representative concepts vectors for documents in the task of text classification. Their idea is similar to ours when building the term-concepts table but without applying the boosting algorithm. In our work, we attempt to leverage the abundant information present in Wikipedia by extracting other features such as strong links and categories structure and integrating them in our system to obtain even better performance. Also, the Exact Match concepts matching methodology we implemented here is similar to the method implemented in [14]. However, it primarily differs in that we perform an expansion to the concepts vector by including related concepts which are not explicitly mentioned in the documents and assign different weights to each depending on how related they are to the EM concepts. This expansion leads to improvement in the system accuracy as shown in the evaluation section.

III. EXTRACTING THE FEATURES FROM WIKIPEDIA

Our approach relies on the use of the vast and highly organized human knowledge existing within Wikipedia giving it a major advantage over other approaches using smaller thesauruses such as WordNet or Open Directory Project (ODP). Due to its openness and structure, Wikipedia is not suitable for being used directly as is by machines and its content needs to be analyzed first with semantic processing tools. In our semantics-extraction system, we treat each Wikipedia article as a unique Concept and use its title as a label. The content of the article is used to help build a relationship vector between the article terms and its title. The formed term-concept vector along with the Categories structure existing within Wikipedia are analyzed to aid in computing the relatedness score between any two text fragments.

There are a number of stages the framework system has to go through before generating the Wikipedia-based vectors. First, we preprocess the available Wikipedia data to retain its articles text, titles, redirect and internal links, and categories structure and remove non-relevant information such as the edit history, image descriptions, and the articles authors. Afterwards, we apply some filtering metrics to extract the important strong links and concepts along with their categories. We then analyze the extracted information and form the term-concepts, concepts and categories vectors. The following sections describe further these elements.

A. Term-Concepts Table

In Wikipedia, each article discusses a sole topic and is uniquely represented with its title. The TFIDF weight of each term existing within the article text is computed. After computing the weights for all the terms in each Wikipedia article, we create a concepts vector for each term. The concepts vectors contain a list of the article titles that contain the term in a descending order based on the weight of the term in each article. We thus call this vector the term-concepts vector. One can view the resulted structure in the form of a non-negative weight table or a sparse matrix where rows correspond to terms and columns correspond to concepts. The weights are computed by using the common TFIDF metric with the following variant:

$$Weight(t,c) = tf(t,c) * \log(\frac{n}{df(t)}) \qquad (1)$$

Where *tf(t,c)* is the term frequency of the term *t* in the article (or concept) *c*, *n* is the total number of articles in the evaluation set, and *df(t)* is the number of articles containing the term *t*. The weight is then l2-normalized to account for the different lengths of the Wikipedia articles.

After building the term-concepts table, we apply a 2-level boosting process which boosts the scores for some entries in the table according to the appearance of some terms in the redirect links titles. This boosting process has been described in [15]. The purpose is to first use the non-repeated info in the redirect links to enhance the rank of the most related concepts as in the term *Unhappy* and concept *Depression (mood)*. Another aim is to enhance the ranking of those concepts for a term *t* that contain that term in their titles. This is evident for the *Unhappy* term and the *Unhappy Triad* concept which appeared higher in its list after the boost.

B. Categories

Each concept in Wikipedia is linked to a parent category or more signifying a form of relatedness between the concept and the categories. Therefore, for each concept we define a vector of the parent categories the concept belongs to. Also, we define other vectors to store the relationship between the categories with each other, in particular the parent/child relationships. These categories vectors are used either directly in one of the heuristics we implemented, or indirectly to aid define the strong links.

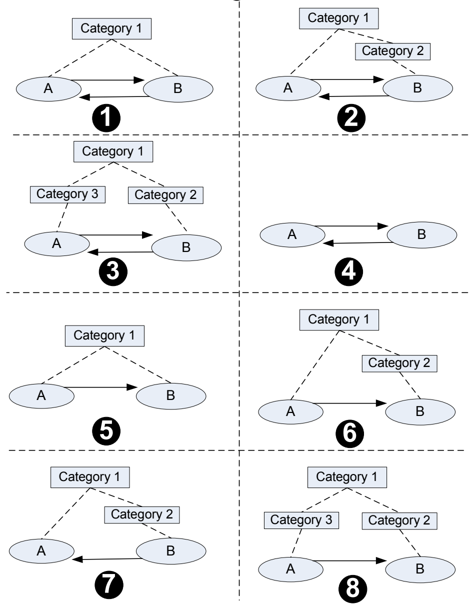


Figure 1. Sorted Link Types based on their weights in descending order

C. Strong Links

The internal links within Wikipedia resemble a form of relationship in many cases between the articles. However, not all the links are of the same significance. We hereby extend the work in [15] and classify the internal links into eight different types. The structure of the Wikipedia categories is employed in defining several of these link types. These link types along with their weights are illustrated in Figure 1. Types 9 and 10 refer to the *See Also links* and the *Inverse See Also* links respectively. We use these link types to form a list between each article and other related articles. Each association is assigned a score based on the weight of the links found that led to forming the association.

Figure 2. Weights assigned for the different links types

w_1	3	**w_2**	2.75	**w_3**	2.5	**w_4**	2.25	**w_5**	1.75
w_6	1.5	**w_7**	1.45	**w_8**	1.25	**w_9**	3.75	**w_{10}**	3.25

IV. Using the Wikipedia-Extracted Features

After preprocessing the documents, we apply a set of heuristics focusing on different aspects of the input documents. The goal is to provide the top recommended classes for TD with the help of the best or all of these heuristics. For each class, we use the documents that belong to it as the basis for extracting representative features of that class. These features are represented as vectors in most of the applied heuristics and are evaluated according to the evaluation policy of the heuristic that generated the features. The scores of all evaluation vectors are aggregated at the end when computing the similarity between the classes' vectors and the TD vector. In the following sections, the extracted features are described along with the process applied to each.

A. Important Terms Extraction

For each class *C*, we have a group of training documents d_i belonging to that class. The idea is to extract a list of the most important terms ordered by their weights to represent each class. The aim is to have representative terms that best distinguish each class from the rest of the text. This is similar to many other approaches previously implemented in different applications including documents clustering [14], cluster labeling [16] and automatic text summarization [17]. In this work, we chose to implement and use it as one of the baseline methods during the evaluation comparisons.

We extract the important terms in each class and give each a score using a weight scheme similar to the described in [16] and [7]. In the implemented metric, inter-class, inner-document and inter-document features are used to compute a weight for each term in each class. They are then grouped in a set for each class to form a representative centroid. For a term *t* in a class *C*, the term weight *w(t,C)* is derived using:

$$w(t,C) = ctf(t,C) \cdot \log(\frac{|C|}{CF_t}) \cdot idf(t,C) \tag{3}$$

Where *ctf(t,C)* refers to the class term frequency and *idf(t,C)* is the inverse document frequency. The inner part of the equation (3) is for the inverse class frequency and attempts to capture the different distribution of a term within the different classes. Intuitively, a term that appears in one class is deemed to be important for that class. Similarly, if a term appears in all of the classes, its discriminative power wouldn't be that important for text classification. As for the *TD*, we simply compute the TFIDF weight for each term. During evaluation, the formed TD vector is compared against the vectors of all classes using the cosine similarity measure. A ranked list of the recommended classes is then generated solely on the basis of the obtained similarity scores.

B. Concepts Extraction

The documents that belong to each class can be represented in the form of a concepts vector. This is usually achieved by devising a mapping scheme that uses the class documents contents to decide which concepts they are mostly related to. Previous approaches have been considered for achieving this. For example, [18] defined some Wikipedia relations between the articles such as synonymy and polysemy and used them with the categories structure to aid in mapping any text fragment to concepts. In [14], the authors created a Wikipedia term-concept matrix from the content of the Wikipedia articles and used it as a bridge for the mapping. In our work, we employ different heuristics for this task utilizing the term-concepts table, the previously defined strong links and the concepts titles.

1) Concepts Centroid

In the first heuristic, we use the term-concepts table to form a vector of concepts for each document in a class. Afterwards, all of the concepts vectors in a class are aggregated and a centroid is created for each. This process has been evaluated in a previous work for the application of Word Sense Disambiguation (WSD) [15]. When grouping the generated concepts for different adjacent terms, the aggregation helps boost the multi-word concepts describing the contexts of the targeted text. However, this is also not without a limitation as large segments of text usually produce noise. In addition, important concepts which are briefly mentioned in a relatively small number of sentences may not appear in the concepts list. We therefore treat each sentence as a separate text fragment and generate a concepts vector for each. We keep only a K maximum of the top representative concepts for each context before grouping them altogether in a document. This way, important concepts which are mentioned once or twice in the document still appear in the document's concepts vector.

2) Strong Links

Expansion through the use of Strong Links was another method implemented in [15]. Some form of relationship between articles can be deduced from their internal links. We used the strong links to aid in finding related concepts for each document and built a concepts vector. Just like with the previous method, we aggregate all of the formed documents concepts vectors to create a representative centroid for each class. The centroid would contain a ranked list of the important concepts along with the score assigned to each. After obtaining the concepts vector for each context, we limit the actual number of concepts to a maximum of L during the evaluation.

3) Exact Match and Related Concepts

In another heuristic we applied, we used a revised Exact Match (EM) approach that is most similar to the one suggested in [14] to find concepts in a document. The idea is to construct the concepts vector by finding the explicitly mentioned concepts within each document in a class. For a text fragment that covers more than one concept using the same terms, we only keep the concept with the largest number of terms. For example, each term in "cat fish", namely cat and fish, is a title for a concept. However, because there is a third concept titled Cat fish that spans all the words, we only choose that concept.

After finding all of the EM concepts in a document and adding them to the list, we choose the 4 most important EM concepts and extend them with a maximum of 10 most related concepts. We use the strong links method previously defined to find the most related concepts to an EM concept. The weight assigned to each EM concept is simply its frequency multiplied by a constant (chosen as 4 in our experiment). As for the weight of a related concept, it would be its frequency multiplied by the actual weight given in Table 2 which depends on how strong the association is. This approach is enhanced with the redirect links to find alternative names or abbreviations for some concepts. The concepts lists for each document would then contain two types of concepts: EM concepts and related concepts. When computing the similarity between a class concepts list and a TD concepts list, the cosine similarity measure is used.

As mentioned above, only the four most important EM concepts are expanded to include related concepts. We use two metrics to help decide which of the found EM concepts in a document are the 4 most important. First, we take into account the frequency of EM concepts in the document. Afterwards, we consider the similarity between the concepts text content (the original text of the concepts from Wikipedia) and the document they appeared in. For this, we use the TFIDF weight and the cosine similarity to compute the similarity between the EM concepts original text and the documents they appeared in.

When an ambiguous EM concept is encountered in a document, we attempt to find its exact meaning by using the centroid based methodology outlined in [15] but here taking into account the class label, the EM concept and its surrounding words. We apply the term-concepts table to expand all the terms in the class label, the EM concept and its 4 surrounding words. This results in several concepts vectors which, after aggregation, form a single sorted concepts vector. The occurrence of the EM concept in the list with the highest score should resemble the disambiguated meaning.

C. *Categories Mapping*

A categories vector is generated for each class after forming the concepts vectors. The previously extracted categories structure from Wikipedia is utilized for this task. After obtaining the concepts vectors with a score associated to each concept, we form a new similar vector in which the concepts are replaced with the categories they belong to. When several concepts share the same parent category, the concepts scores are aggregated into one and then it is associated with the shared parent category in the categories vector. This aggregation in turn signifies the dominant categories which are parents to the largest number of concepts in a class. For each class, we limit the number of obtained categories to M. The same process is applied to the TD too to generate a categories vector.

D. *Classifying a Test Document*

In the previous sections, we described the different stages involved in obtaining the important terms from each class and enriching its documents content with Wikipedia-extracted features including concepts and categories vectors. We highlight here how to classify a TD and combine the previously defined similarity measures into one. This is achieved by computing the aggregated score *gs(TD, CL)* for the test document *TD* and the class *CL* as follows:

$$gs(TD,CL)=\sum_{i=1}^{q}\beta_i Sim_i(TD,CL) \tag{4}$$

Where q is the total number of heuristics or methods we apply, and β is the weight we give to the different heuristics we implement. The sum of all β_i in the above formula is one and thus each β_i is a number between zero and one. In this work, we experimented with different values of β in the evaluation section as we deemed reasonable and reported their results.

V. EXPERIMENTS AND EVALUATION

A. Dataset

We performed classification experiments on two data collections. The first is the 20 News Groups (20N) which has newsgroups documents that were categorized manually into 20 different classes. Each class has nearly 1,000 documents making the total of documents available in the collection almost 20,000. The second collection was built from the ODP. We selected 100 random categories and downloaded 100 pages for each category. The total number of documents is 10,000. Both datasets are balanced in the sense that the number of contained documents in every category is almost the same.

To both collections, all documents are parsed and tokenized. For the 20N collection only the title, subject, keywords and content are kept. Also, we keep only the titles and the content of the downloaded pages for ODP. We computed the TFIDF weight for each term. Terms appearing in the titles and the keywords section have their weights doubled to emphasize their importance. Unless otherwise stated in one of the applied heuristics, the title and the keyword sections are treated as part of the content of the articles.

We divided each dataset into five random, but equivalently sized sub-datasets retaining the documents distribution balance in each category. We use the five sub-datasets to evaluate our methods five times. Then, we take the average of the obtained accuracies as the final answer.

B. Methods and Evaluation Setup

We conducted several experiments to evaluate the different methods and heuristics suggested in this work. Specifically, we compared the performance of the following approaches:

- TFIDF: A centroid is formed for each class using the common TFIDF weights of its documents terms.
- ITC: Important Terms Centroid is created for each class based solely on the important terms in the class documents. Equation (3) is used to compute the terms weight in the centroids and TFIDF is used for the TD. Cosine similarity is employed for comparisons.
- CC-Rel: Concepts Centroid is created for each class. The concepts vector is constructed by expanding the documents terms with the boosted term-concepts table as described in section 4.B.1.
- CC-Rel-Ex: Concepts centroid is constructed for each class with the help of the strong links as described in section 4.B.2. For this, we apply the term-concepts table first to form a concepts vector just like in CC-Rel. Then, we expand the concepts list by including related concepts with the help of the Strong Links method.
- CC-EM: Concepts Centroid is constructed for each class by finding the concepts which are explicitly mentioned in the documents. For each document, we choose only the top 5 concepts.
- CC-EM-Ex: Concepts Centroid is constructed for each class with the help of strong links as described in section 4.B.3. The main difference between this and CC-Rel-Ex is in how the initial concepts vector is constructed. In this method, we detect the explicitly mentioned concepts and use them to build the initial list.
- G: Categories centroid is created for each class by aggregating the categories of the encountered concepts using the best CC method.
- ITC-CCRelEx: This forms vectors for the important terms in each class, and the concepts using CC-Rel-Ex method.
- ITC-CCEMEx: This forms two vectors for each class, namely Important Terms vector and EM Concepts extended with related concepts using the strong links method.
- ITC-G: Important terms, and categories centroids are created for each class. The categories vector is implemented using the method described in section 4.B.4. The categories are formed from the concepts of the approach CC-Rel-Ex, though the concepts vectors are not used directly when computing the similarity between the different vectors.
- ITC-CCRelEx-G: Important Terms, Concepts and Categories Centroids are created. The categories vector is implemented using the method described in section 4.B.4. The concepts vector is constructed as in CC-EM-Ex.

For all the methods that utilize the term-concepts table, we applied the boosting to the table with the parameters FLBS and SLBS assigned to 1 and 1.05 respectively which we found to be reasonable at the time. When Forming the Concepts centroid, we chose 20 for K as the maximum number of concepts to be obtained for any term from the term-concepts table. As for finding related concepts using the strong links method, 10 for L was the maximum number of related concepts extracted from any document. Equation (4) takes part in the implementation of the approaches labeled ITC-CCRelEx, ITC-CCEMEx, ITC-G, ITC-CCRelEx-G, and ITC-CCEMEx-G. For these approaches different values of β has been assigned and the results we display are those given by the best combination. The evaluation accuracy for a run is measured by dividing the number of correctly categorized documents over the number of total documents in a class. As for the categories labels of the 20N, they were updated display clear form of words.

C. Results

The results we obtained for the run experiments are show in Table 3. ITC-CCRelEx-G is the best among all for both datasets. For the 20N dataset, the β variable was assigned {0.3, 0.5, 0.2} for the important terms vector, concepts vector and categories vector, respectively. As for the ODP collection data, they were chosen to be {0.3 , 0.4, 0.4}. For both collections, the expirment CC-Rel-Ex, which uses the term-concepts table to locate concepts in a document and then extend it with related concepts using the strong links method, yielded better results than CC-EM-Ex. Also, extending the located concepts in documents with other related ones not explicitly mentioned in the document proved to be useful. This was especially evident for the runs CC-Rel and CC-EM giving a performance increase of up to %4.3. The second best method for both data collections was for ITC-CCRelEx and the weight β assigned to the important terms vector and the concepts vector was {0.4,0.6} for both collections.

TABLE I. ACCURACY RESULTS OBTAINED FOR DIFFERENT VARIATIONS OF THE SYSTEM

	20N	**ODP**
TFIDF	82.98	69.26
ITC	84.78	71.54
CC-Rel	85.82	74.85
CC-Rel-Ex	87.62	77.63
CC-EM	84.84	74.06
CC-EM-Ex	86.79	77.24
G	87.08	77.73
ITC-CCRelEx	87.97	78.33
ITC-CCEMEx	86.94	77.12
ITC-G	87.71	77.72
ITC-CCRelEx-G	**88.63**	**78.82**
ITC-CCEMEx-G	87.18	77.31

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a system incorporating background knowledge in Wikipedia to enrich the representation of text documents and enhance the task of Text Classification. We extracted different sets of features from Wikipedia including term-concepts table, categories structure and strong links and organized them for subsequent processing. We then investigated different sets of heuristics employing the extracted features from wikipedia and reported their results. Accuracy improvements of up to 13.8% were achieved when compared against the common BOW approaches. We plan to apply and optimize our methodologies for work on other applications such as Text Summarization and hierarchical classification in our future work. We also will investigate the use Wiktionary and in particular its Words relations to extend the work applied here. Finding a methodology to automatically obtain the optimal parameters for the different methods we outlined is also something that we need to work on.

REFERENCES

[1] B.J. Jansen, D.L. Booth, and A. Spink, “Determining the user intent of web search engine queries,” *Proceedings of the 16th international conference on World Wide Web*, Banff, Alberta, Canada: ACM, 2007, pp. 1149-1150.

[2] S. Dumais and H. Chen, “Hierarchical classification of Web content,” *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, Athens, Greece: ACM, 2000, pp. 256-263.

[3] S.R. Gunn, “Support Vector Machines for Classification and Regression,” 1998.

[4] R.D. Goyal, “Knowledge Based Neural Network for Text Classification,” *Proceedings of the 2007 IEEE International Conference on Granular Computing*, IEEE Computer Society, 2007, p. 542.

[5] L. Baoli, L. Qin, and Y. Shiwen, “An adaptive k-nearest neighbor text categorization strategy,” *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 3, 2004, pp. 215-226.

[6] S. Eyheramendy, D.D. Lewis, and D. Madigan, “On the Naive Bayes Model for Text Categorization,” 2003.

[7] H. Guan, J. Zhou, and M. Guo, “A class-feature-centroid classifier for text categorization,” *Proceedings of the 18th international conference on World wide web*, Madrid, Spain: ACM, 2009, pp. 201-210.

[8] Z. Cataltepe and E. Aygun, “An Improvement of Centroid-Based Classification Algorithm for Text Classification,” *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, IEEE Computer Society, 2007, pp. 952-956.

[9] M.D.B. Rodriguez, J.M.G. Hidalgo, and B.D. Agudo, “Using WordNet to Complement Training Information in Text Categorization,” *cmp-lg/9709007*, Sep. 1997.

[10] K. Dave, S. Lawrence, and D.M. Pennock, “Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews,” 2003, pp. 519--528.

[11] D. Milne, “Computing Semantic Relatedness using Wikipedia Link Structure,” *Proceedings of the New Zealand Computer Science Research Student Conference*, 2007.

[12] “Overcoming the Brittleness Bottleneck using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge.”

[13] E. Gabrilovich and S. Markovitch, “Computing semantic relatedness using Wikipedia-based explicit semantic analysis,” *IN PROCEEDINGS OF THE 20TH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, 2007, pp. 1606--1611.

[14] X. Hu, X. Zhang, C. Lu, E.K. Park, and X. Zhou, “Exploiting Wikipedia as external knowledge for document clustering,” *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Paris, France: ACM, 2009, pp. 389-396.

[15] M.O. Abdullah Bawakid, “Using Features Extracted from Wikipedia for the Task of Word Sense Disambiguation,” *9th IEEE International Conference on Cybernetic Intelligent Systems 2010*, Reading, UK: IEEE, 2010.

[16] D. Carmel, H. Roitman, and N. Zwerdling, “Enhancing cluster labeling using wikipedia,” *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, Boston, MA, USA: ACM, 2009, pp. 139-146.

[17] A. Bawakid and M. Oussalah, “A semantic summarization system: University of Birmingham at TAC 2008,” *Proceedings of the First Text Analysis Conference (TAC 2008)*, 2008.

[18] J. Hu, L. Fang, Y. Cao, H. Zeng, H. Li, Q. Yang, and Z. Chen, “Enhancing text clustering by leveraging Wikipedia semantics,” *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, Singapore, Singapore: ACM, 2008, pp. 179-186.

A Semantic-Based Text Classification System

Abdullah Bawakid and Mourad Oussalah

School of Engineering
Department of Electronic, Electrical and Computer Engineering
University of Birmingham
{axb517 , M.oussalah}@bham.ac.uk

Abstract

This paper presents a system that performs automatic semantic-based text categorization. Using Princeton WordNet, a series of induced methods were implemented that extract semantic features from text and utilize them to decide how similar a document is to different topics. In addition, a bag-of-words method incorporating no knowledge from WordNet is implemented in the system as a basis to compare different WordNet-based approaches. This paper describes the system and reports on a simple analysis performed to evaluate the different implemented methods. At the end, a discussion on the limitations of this study and the future work to optimize the system is presented.

Keywords: Semantic Similarity, WordNet, Categorization, Text Classification, Information Retrieval, Word Sense Disambiguation

1. Introduction

Most of the available data nowadays exist in the form of electronic unstructured text information. The size of the data and its rapid growth makes it challenging for users to analyze, organize and access the required information efficiently. Therefore, document categorization, as an efficient tool that enables effective retrieval, organization and summarization tasks, has received more and more attention from information processing community as suggested by the impressive number of publication in the field, e. g., [14]. Typically, a text classification, or categorization, is the task of automatically assigning a class label to an unlabelled document where the set of classes may either be previously defined, or automatically generated from the document.

During the last decades, a large number of text categorization systems have been proposed using a variety of approaches such as support vector machines [4], boosting algorithms [13], term frequency and inverse document frequency [16], among others. Most of these systems use the bag-of-words model or vector space representation by having individual words (or word stems) as basic representative features for the document content [8, 12].

While bag-of-words approaches present a good performance on many machine-learning tasks due to low computational cost and inherent parallelism, their limitations are also well acknowledged. Especially, the underlying classification scheme is restricted to detecting patterns within the used terminology only, which excludes conceptual patterns as well as any semantically related words. It is thus possible to gain better results across multiple domains by utilizing an external semantic thesaurus like WordNet that defines an upper-level of relationships among most of the terms in the testing data. Typically, WordNet is a database for the English language containing semantic lexicon that organizes words into groups of synsets [7]. Every synset stands for a single word prototype that refers to a group of words that share the same (semantic) meaning. In addition to making use of relations in WordNet, features such as Part-of-Speech tags have been considered in [8]. This motivates the intensive research carried out in this issue which had given rise to a variety of implemented systems incorporating features derived from the common semantic thesaurus WordNet and its words relations [2, 6, 11]. Strictly speaking, the hierarchical organization of WordNet involves important distinction between various part of speech (PoS) parts. Indeed, while categorization of nouns into underlying taxonomies, headed by a unique beginner such as animate or artifact is straightforward, this does not extend to verbs, which are rather partitioned into several semantic fields with many overlapping [7]. This discrepancy between verbs and nouns obviously influences the calculus of semantic similarity, especially when dealing with sentences where the word-by word semantic similarity has proven usually to be non-effective [5], which, in turn, influences negatively the performance of retrieval, summarization and categorization tasks. This makes the debate of nouns versus verbs semantic similarity widely open. This paper attempts to contribute to this debate by investigating the influence of six different schemes on the performance of text categorization. This consists of co-occurrence frequency, expansion with synonyms, PoS semantic similarity, Noun-semantic similarity, semantic similarity with verb-noun conversions and semantic similarity with adverb/adjective conversion. The proposals were implemented in Java platform making use of WordNet libraries and Jiang's semantic similarity measure [3]. A test data constructed from news website is used to evaluate the performance of the various schemes. The developed system performs automatic text classification through semantic-based approaches relying on semantic similarities between the testing data and the classes where, for each class, a score that quantifies the closeness of a given document with the underlying class is computed. The document is deemed to belong to a specific category if the latter endows the highest score.

Section 2 presents an overview of our system, its main stages and how it works. In section 3 we discuss the evaluation tests performed on different runs for the implemented methods in the system. While discussions and perspective works are given in Section 4.

2. System Overview

Figure 1 shows the main steps performed in the preprocessing stage of the developed system. Overall, tokenization, filtering and stop words removal were applied to both the categories and the test documents. More formally, the text documents are cleaned from any unnecessary information such as XML/HTML tags. Some of the category names are merely one or two phrases while others are two complete sentences or longer. Because of these variations, a sentences splitter is applied to both the categories and the document that needs to be classified. Tokenizing the words and converting them to their morphological forms is the next step (with the help of WordNet [7]). Tagging the words with their Part-of-Speech, which distinguishes verbs, nouns, adverbs/adjectives, among others, is the last step in the preprocessing stage.

After preprocessing both the categories and the document, the classification process takes place. Each category or class is given a score that quantifies the extent to which the class is close to the given document. The class that has the highest score is the one assumed to represent the document. Various classifiers were implemented for the purpose of quantifying the document-category matching based on the way the semantic similarity is calculated, if any. In each of these variations, we compare the terms comprising the class with the document terms (or the induced terms). The scores of all classes are initially zeros. If there is a *hit*, which is meeting the condition defined in the system variations, the score of the category increases by one. Thereby, the category achieving high score will inherent its label to the document.

2.1 Co-occurrence frequency

This is referred to as a Base classifier. It is a simple classifier that does not use WordNet or any of its features. Instead, a *bag-of-words* representing the document with the frequency of each word is compiled which then is compared to all the terms in each category. The class having the largest number of co-occurring terms in the bag is considered the one that the document belongs to. The words in the bag are given different weights depending on their number of occurrences in the document. POS Tagging is not needed in the preprocessing stage for this variation of the classifier. For example, given the document "Sam likes playing football", and the two classes "football playing" and "Surgery", the score of the first class is 2 (co-occurrence of two words –"football" and "playing"-) and the second is 0 (no co-occurrence).

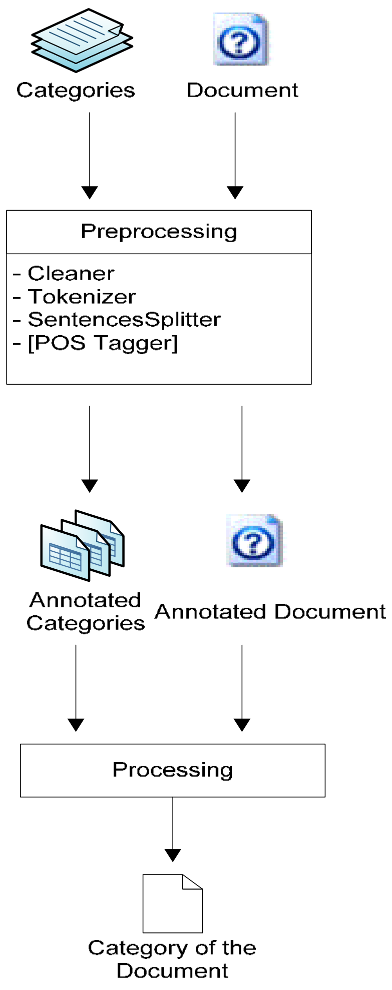


Figure 1 The Overall System Architecture

2.2 Expansion with Synonyms

The document terms are expanded with their synonyms using WordNet. A simple words-matching then takes place between the tokens in each category and document terms. The category that has the largest number of matches with the document terms is considered to be the one the document belongs to. For example, the sample document "Alan had flu" (Doc2 in Figure 2 after removing the stop words) would be expanded to become "Alan influenza flu grippe" as in Figure 3. The expanded document is then compared with the terms in each class. The standard Co-occurrence frequency approach is then applied to the synonym-expanded documents; that is, when a match is found, it's considered a *hit* and the score of the class is increased by one.

2.3 Semantic Similarity

With this approach, the semantic similarity score is computed between document nouns and category nouns, and document verbs and category verbs. Using the semantic similarity measure Sim_{jcn} [3], similarity scores are computed between each term in the document, and all terms in the classes of the same part-of-speech. More specifically, given two words: w1 belonging to the category text, and w2 belonging to a document, the semantic similarity score is computed as follows:

$$Sim_{jcn}(w1, w2) = (ic_{res}(w1) + ic_{res}(w2)) - 2 \times sim_{res}(w1, w2)$$

Where IC stands for Information Content, or the number of times a term occurs in a corpus and is expressed in the form of a probability. The formula used for measuring IC is:

$$ic_{res}(w) = -\log P(w)$$

Where P(w) is the probability of encountering a word w in a corpus, which is quantified using frequency-based approach. The formula for Sim_{jcn} is:

$$Sim_{res}(w1, w2) = \max_{w \in S(w1,w2)} ic_{res}(w)$$

In our system, we used an information content file provided with the WordNet::Similarity package [9]. Although there are several IC files provided, we have limited our analysis to ic-bnc-resnik-add1 which is based on the British National Corpus, world edition released in 2000 and is based on the Resnik Counting [10].

After obtaining the similarity score, if it is found to be equal to (or bigger than) a predefined threshold, it is a *hit*. The document would then belong to the category that has the largest number of hits.

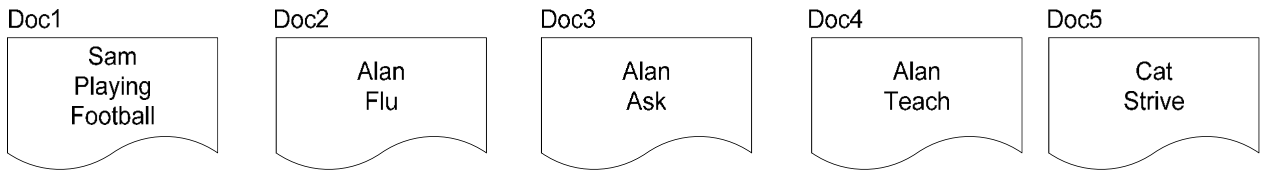


Figure 2. Sample of documents in the processing stage

Syn(Flu) = {Influenza , flu, grippe}
Syn(Ask)= {inquire, enquire, require, expect, necessitate, postulate, take, involve, demand, call_for}
NounOf(Teach)= {teacher, teaching, instruction, instructor}
NounOf(Ask)= {asker, inquirer, inquiry, inquiring, enquirer, enquiry}
NounOf(Strive)={endeavor, endeavour, striving, strain}

Figure 3. Finding Nouns and synonyms of different words through the use of WordNet Relations

2.4 Noun-Semantic Similarity

As in the previous approach, Jiang's semantic measure is used with this one, too. However, the words to be compared with this variation are always nouns, while verbs, adjectives and adverbs are omitted. If the semantic similarity score is found to be larger than a previously set threshold, it is a hit. The document would, again, belong to the category that has the largest number of hits. For example, the term "Strive" that appears in Figure3 Doc5 would be omitted with this approach.

The reason for omitting the evaluation of verbs, as will appear in the experiment section, is to evaluate whether the next approach adds noise to the nouns-only method, makes no changes, or improve the results as desired.

2.5 Semantic Similarity with Verb-Noun Conversion

With this approach, all verbs are converted into nouns using WordNet relations whenever possible. The hypothesis here is that "a higher accuracy will be achieved when computing the semantic similarity score between two sentences or phrases, if all words can be converted to a form that allows them to be all compared with each other". Since WordNet contains more Nouns than verbs, it was decided to convert verbs into nouns, and not the opposite, using the different WordNet relations such as *derivational* links, *synonyms* and *Pertainyms*. A custom function was developed to perform the conversion operation. It starts by checking if one of the derivations for one of the verb senses yields a noun. If not, the synonyms of the verb are explored and we try to find whether one of the derivations for the synonyms gives a noun. If not successful, we explore the *Pertainyms* relations along with the related adjectives and

adverbs which are one level close to the verb, and then we move one more level if a noun is not found.
By applying the above process, the first encountered noun is used as a representative for the verb. It should be noted here that not much analysis was performed on Words Senses Disambiguation due to the lack of an intelligent technique that could be applied in our setup and for our application.
Few examples illustrating the implemented method are illustrated in figure 2. The verb "ask" for example is replaced with the nouns "asker, inquirer, inquiry, inquiring, enquirer, enquiry". When performing the comparison, if any of these nouns give a semantic similarity score equals or above the desired threshold, it is a *hit*.

2.6 Semantic Similarity with Adverbs/Adjectives Conversion

In contrast to previous approach, we convert any encountered adjective or adverb in the processed document into their corresponding nouns or verbs whenever possible. The limitation arises because there are some adjectives/adverbs in which the conversion is not possible. We use the different relations in WordNet for achieving this task including Pertainyms, synonyms, antonyms, and derivational links. For example, the adjective "American" is transferred to the noun "America". After performing the conversion, we compute the semantic similarity of the nouns (original nouns and converted nouns) from the documents with nouns in each class. We do the same to verbs. For each comparison when the semantic similarity score is above the desired threshold, it is a hit.

3. Experiment

The classifiers mentioned in Section 2 have been implemented in Java platform. Furthermore, the system provides some flexibility to the user to configure the weight to different parts of the documents. For instance, more weight might be assigned to the first portion of the document that includes the abstract or last part, which likely includes the conclusion of the document. Similarly, the threshold associated to the semantic similarity and governing the hit part can also be adjusted by the user. Indeed, after computing the semantic similarity score, we compare its value with a predefined threshold. We say there is a hit only if the score is equal to or above that threshold. To improve the overall performance of the system, a semantic similarity matrix between all the nouns in WordNet was constructed. The software runs on an IBM-compatible PC with a 3.0 GHz Pentium 4 processor and 4 GB of memory. The semantic similarity matrix was built with the help of PJWSL[1]. Gate [2] ANNI Tagger was used for identifying the POS of words.

3.1 Test Data and Metrics:

35 documents which were previously manually classified by humans were used as the testing data. The documents are scientific papers covering different topics that were classified based on their domains. The total number of domains, or classes, that they were classified to is 25. The system was run with the different variations mentioned above. Different thresholds were used for the semantic similarity-based runs: 0.1, 0.2, 0.3, 0.45, 0.5, 0.6, 0.7 and 0.8.

Method	Base	Syn	Sim	Sem Sim (N)	Sem Sim (VC)	Sem Sim (AAC)
Accuracy %	85.5	88.8	87.4 (at 0.7)	90.3 (at 0.7)	84.2 (at 0.7)	87.4 (at 0.7)

Table 1: Evaluation results of the accuracy of the different variations of the system

The accuracy is measured by dividing the number of right classifications by the total number of documents
If a document was found to belong to two different categories, and one of the two categories is correct, then it is assumed that the classification is correct. A summary of the results of the runs is shown in Table 1.
For each variation, Syn, which stands for the synonyms-expansion method in the table, does not require the use of the threshold. Sim refers to the semantic similarity-based method. NO refers to Nouns-Only and VC refers to the Verbs-Conversion version of the system. AAC corresponds to the method involving adjectives/adverbs conversions. Sim, NO and VC all require setting the threshold value before each run. The table illustrates the best-obtained accuracy score along with the threshold it was obtained at.

3.2 Results

As shown in Table 1, the introduction of WordNet-based features to the system did make significant changes. The accuracy obtained for the base system is 85.5%. The Synonyms method seems to have a relative increase of 3.3 over the base accuracy.

As for the rest of the methods, choosing different thresholds when deciding about whether the similarity score for a pair of words is a hit or not did make significant differences. By examining and comparing the results in Table 1, it seems that evaluating only nouns when forming the pairs for computing the semantic similarity yields the best result, given that the right threshold is chosen. The semantic similarity method which is solely based on nouns, without conversions, gave the best overall accuracy: %90.3.

[1] Pure Java WordNet Similarity Library by Mark GreenWood
http://nlp.shef.ac.uk/result/software.html
[2] Gate: General Architecture for Text Engineering, University of Sheffield

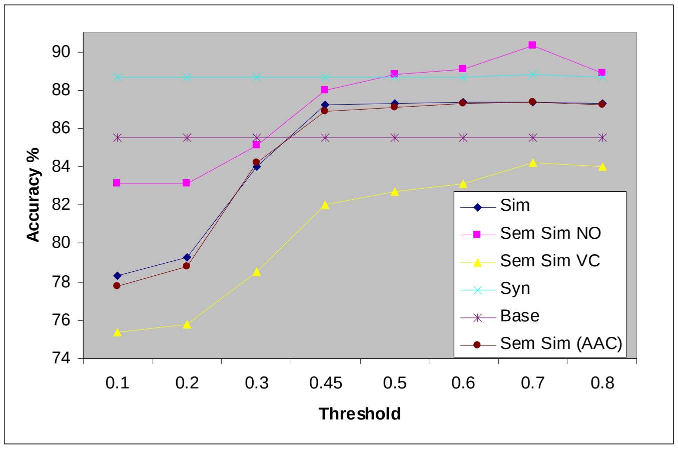


Figure 4: Illustration of the effects of changing the thresholds at difference runs of the system

An illustration of the effects of changing the threshold value for different runs of the system on different method is shown in figure 4. It can be noted that for the methods relying on semantic similarity computations, setting the threshold to a low value (generally below 0.45), creates a lot of noise and thus decrease the accuracy of the system. The best results from each method of the mentioned method were obtained when the threshold value was set to 0.7. Increasing the threshold value beyond that had either a negative effect on the result or no effect at all. All of the methods, except for the VC, resulted in better results than the base when the threshold was set to values bigger than 0.3.

3.3 Discussion:

While it does seem from the results that the introduction of WordNet-based semantic similarity features gave a positive effect, the system still needs to get the right threshold for optimum results. There are other factors which should be considered as well such as the number of documents being evaluated and the domains they were taken from. While the authors have tried to choose documents from domains which are as much diverse from each other as possible, it is expected that running the system on a large corpus with larger number of documents per domain will yield more accurate results.

It is also noted that the function that has been developed for converting verbs into nouns using WordNet relations did not perform as desired. It is possible that this was caused by not setting a limit to the number of *relations* a conversion may require going through before generating a noun in some cases which can lead to unexpected results. One could have tackled this by introducing a penalty affecting the semantic similarity score. The deeper the relations the conversion process had to traverse through, the less weight the computed score is given.

Even though WordNet gave better results in our experiment and others too [6], using it as the main backend source to classify documents may not be optimum. WordNet was manually built and covers a limited number of concepts that may not include some aspects in many domains. Furthermore, the WordNet ontology is aging and costly to update and assemble again and does not include facts such as the current president of the United States. Again, this is mainly because it is manually assembled. It is therefore useful to consider using other thesaurus such as Wikipedia which is constantly being updated and is the world largest encyclopedia. For example, the number of synsets existing in WordNet 3 is 115,000 while there are over 2 million articles in Wikipedia which can each be treated as a concept. Many of these concepts contain multiple words in their names which are not found in WordNet. Having a larger number of concepts covered in the system's underlying reference ontology in addition to handling multi-word concepts is expected to boost the performance of the system tremendously.

In addition, the availability of articles in different languages opens the possibility of investigating languages other than English and comparing the performance of the implemented methods in different or similar domains from different languages.

4. Future Work

We plan to add and improve many aspects of the system we developed, especially in the processing part. Among the ideas we plan to integrate are the following:

- The current system focuses on terms and single words alone, and not multi-word concepts. Sometimes, a combination of two words may carry a different meaning from each word alone, as in "Crowd Financing". Using a larger thesaurus than WordNet such as Wikipedia [15] or multi-word units when scanning documents (windows) [1] are two approaches that will be explored.
- Finding a method to automatically optimize the weight of the dynamic features. Currently, the thresholds are assigned manually based on the user's observations.
- Handling polysemous (one term carrying multiple meanings) and synonymous words. If a word has multiple senses and is to be compared with another word, our system currently chooses the highest similarity score.
- The lack of an intelligent scheme for Word Sense Disambiguation may have been the cause for the decrease of the performance of the Verbs-Conversion approach described above.

5. Conclusion

In this paper we presented our on-going work on building an automatic semantic-based classification system and the results of the experiments we had. The results suggest that using WordNet-based semantic approaches does yield to a better accuracy given that the right parameters (i.e. semantic similarity threshold) are selected. In future work, we plan to apply and experiment with more detailed measures to handle different aspects such as automatic optimization for the dynamic features, polysemous/synonymous words and the use of the world's largest encyclopedia, Wikipedia

References

1. Bloehdorn, S. and A. Hotho, *Boosting for Text Classification with Semantic Features* Proceedings of the MSW 2004 Workshop at the 10th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2004
2. Jensen, L. and T. Martinez, *Improving Text Classification by Using Conceptual and Contextual Features* Proceedings of the Workshop on Text Mining at the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000.
3. Jiang, J. and D. Conrath, *Semantic Similarity based on Corpus Statistics and Lexical Taxonomy.* Proceedings of International Conference Research on Computational Linguistics,Taiwan, 1997
4. Joachims, T., *Text Categorization with Support Vector Machines: Learning With Many Relevant Features.* Proceedings of ECML, 1998.
5. Li, Y., McLean, D., Bandar, Z. A., O'Shea, J. D., and Crockett, K.,Sentence similarity based on semantic nets and corpus statistics, IEEE Transactions on Knowledgeand Data Engineering Vol 18, No. 8, pp. 1138–1150, 2006.
6. Mansuy, T.N. and R.J. Hilderman, *A Characterization of Wordnet Features in Boolean Models For Text Classification.* AusDM 2006, 2006: p. 103-109.
7. Miller, G., *WordNet: A lexical database for English.* Communications of the ACM, 1995: p. 39-41.
8. Padmaraju, D. and V. Varma, *Applying Lexical Semantics to Improve Text Classification.* Proceedings of Second symposium on Indian Morphology, Phonology and Language Engineering, 2005: p. 94-98.
9. Pedersen, T., S. Patwardhan, and J. Michelizzi, *Wordnet::similarity - Measuring the Relatedness of Concepts.* Proc. of AAAI-04, 2004.
10. Resnik, P., *Using information content to evaluate semantic similarity in a taxonomy.* Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995.
11. Rodriguez, M.d.B., J.M.G. Hidalgo, and B.D. Agudo, *Using WordNet to complement training information in text categorization,.* Second International Conference on Recent Advances in Natural Language Processing, 1997.
12. Salton, G. and M.J. McGill., *Introduction to Modern Information Retrieval.* McGraw-Hill Book Co., 1983.
13. Schapire, R.E. and Y. Singer, *BoosTexter: A Boosting-based System for Text Categorization.* Machine Learning, 2000: p. 135-168.
14. Sebastiani, F., *Machine Learning in Automated Text Categorization.* ACM Computing Surveys, 2002. **34**: p. 1-47.
15. Wang, P. and C. Domeniconi, *Building Semantic Kernels for Text Classification using Wikipedia.* International Conference on Knowledge Discovery and Data Mining 2008: p. 713-721.
16. Yang, Y. and C.G. Chute, *An example-based mapping method for text categorization and retrieval.* ACM Transactions on Information Systems, 1994: p. 252-277.

A Semantic Summarization System: University of Birmingham at TAC 2008

Abdullah Bawakid and Mourad Oussalah

University of Birmingham, School of Engineering
Department of Electronic, Electrical and Computer Engineering
Edgbastoon, Birmingham B15 2TT
{ axb517 , M.oussalah}@bham.ac.uk

Abstract

Text summarization of document or multi-documents has been acknowledged as one of the most challenging tasks in information system community because of the rich semantic structure of the language and the subjectivity inherent to the summarization task. In this paper, a new query-based extractive summary methodology is put forward. The approach makes use of phrasal decomposition of the text where each sentence is ascribed a scoring function, which will then be used to identify the most relevant sentences in the sequel. The scoring function is expressed as a convex combination of a set of features that are extracted beforehand from the (multi) document(s). Besides, the scoring function includes a semantic similarity evaluation where the WordNet taxonomy is used in conjunction with a variety of other extracted features, as a basis to construct the sentence-sentence semantic similarity. The system architecture as well as its linguistics processing parts are described. Finally, we present the results of our participation in TAC 2008 with possible perspectives.

Keywords: Semantic Similarity, WordNet, Linguistic Quantifiers, Text Summarization, Information Retrieval

1. Introduction

Text Summarization, as the process of identifying the most salient information in a document or set of documents (for multi-document summarization) and conveying it in less space, became an active field of research in both Information Retrieval (IR) and Natural Language Processing (NLP) communities. Summarization shares some basic techniques with indexing as both are concerned with identification of the essence of a document. Also, high quality summarization requires sophisticated NLP techniques in order to deal with various Parts Of Speech (POS) taxonomy and inherent subjectivity. Typically, one may distinguish various types of summarizers.

Loosely speaking, most common existing summarizers work in an extractive fashion, where portions of the input documents, for instance, sentences, which believed to be more silent, are selected to form the summary. On the other hand, non-extractive does not rely on text selection but rather on a deeper understanding of input text. Query-based summaries are generated in reference to some user query (e.g., summarize a document about an international summit focusing only on the issues related to the environment)

This paper advocates a trade-off methodology between extractive and query-based summarization. The former is due to the fact that the developed methodology uses a scoring function, which uses WordNet taxonomy to generate sentence-sentence semantic similarity as well as a set of extracted features, to quantify the relevance of each sentence. This yields a resulting summary which is nothing else than the most ranked sentences. While the query-based approach is due to the explicit accounting of the topic-sentence semantic similarity in the overall methodology as it will be detailed later on.

The paper describes the system we developed to participate in the update task of TAC 2008. The update summarization task requires participants to submit fluent and organized 100-word multi-document summaries of a set of news articles under the assumption that the user has already read a given set of articles earlier. The summaries to be generated should be relevant to the topic statement given by the user. The purpose of each summary is to inform the reader of new information about a particular topic. The test documents provided were chosen from the AQUAINT-2 collection[1].

The next section gives some background and relates our work with existing summarization systems. In section 3, we give an overview on our system, its main components and how it works. In section 4 we discuss the evaluation performed by NIST on our submitted runs and the obtained results. In section 5 we present some ideas for future work and how the system can be improved.

[1] See http://www.nist.gov/tac/tracks/2008/summarization/ for the detailed task description.

2. Background

In the past few years, many multi-document summarization systems have been implemented, most of which are extractive. The key in such systems is to extract the most relevant parts from the source to the user. An example for such systems is MEAD [1] [2] which ranks sentences using a linear combination of features and forms summaries from the highest scoring sentences. MASC [3] is another feature-based summarization system that performs compressions to sentences after the extraction stage.

Our system assigns a score to each sentence in the source documents based on a set of static and dynamic features. Static features include sentences locations and the number of Named Entities (NEs) in each sentence. Dynamic features on the other hand are those that change based on the document sets chosen. The score given for the semantic similarity between a sentence, and the rest of the sentences in the documents set is an example of a dynamic feature employed in our system. Part of our system performs analysis on linguistic quantifiers and combines it with the semantic similarity computing module to form a metric affecting the score given to each sentence.

3. System Overview

Figure 1 shows the three main stages involved in generating summaries with our summarizer: *Preprocessing* the source documents, *Extracting and Analyzing* the features, and *Generating the summaries*. The documents are preprocessed first and prepared to extract the features of their sentences.

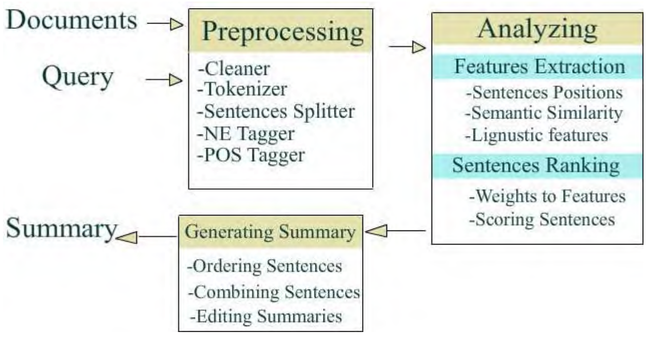


Figure 1 The Summarizer Architecture.

After extracting the features, a score is computed for each sentence based on the extracted features. The summary is presented at the end by iterating through the sentences and selecting the highest-scoring candidates till the maximum number of words is reached.

The TAC 2008 update task requires participants to submit ~100-word summaries given a group of documents and a topic statement (title and narrative). In our system, the topic statement was treated as the user query. The 100-word limit was met by examining the length of the last sentence appearing in the summary. The 100-word limit was met by iterating through all the highest scoring sentences, starting with the highest rank and proceeding with the next lowest ranked and appending them to the summary until the limit is reached or all candidate sentences are exhausted. If the addition of the last sentence in summary caused the summary length to exceed the limit, it is replaced with the next shorter high scoring sentence. This process can be improved by adding a stage for editing the summary to shorten its length by removing unnecessary information from the summary sentences. Due to time constraints, the editing process was not applied. The following sections examine each of the summarization stages in more details.

3.1 Preprocessing

The preprocessing stage involves cleaning the source documents, splitting and annotating the sentences, and extracting the features.

First, unnecessary information and tags are removed from the source documents such as the HTML/XML tags, news agencies names and tables containing numbers. Then, key parts from the documents are extracted such as the publication dates, the documents IDs, and the headlines. The document ID and publication date along with the document name are used to identify each document during the different processing stages. The headline is treated as the document title as explained in the next section. Sentences and word boundaries are then detected and different features are extracted with the help of GATE [4] from the source sentences and the provided user query. The extracted features and annotations include Named Entities in each sentence (Locations, Organizations, and Persons), Part-of-Speech tags (POS), and co reference resolution.

After preprocessing the documents and the queries, the processing stage begins scoring sentences based on the computed/extracted set of features detailed in next section.

3.2 Summarization Features

3.2.1 Sentences Location:

The position of sentences in a document can play a significant factor in finding the sentences that are most related to the topic of the document [5]. So, we have decided to take into account the position of sentences when computing the score for each sentence. More weight is given to sentences at the beginning and end of each document than the rest.

3.2.2 Named Entities:
Using GATE, it was possible to recognize the Named Entities (NEs) mentioned in each document. The sentences containing more NEs are assumed to be more important than those that contain no NEs. Only the frequency of NEs in each sentence and the document was taken into account when forming the scoring formula.

3.2.3 Title / Query
The title of the document, if any, as well as user's query or abstract sentence(s) used to characterize the document or a set of documents are without doubt of paramount importance to quantify the relevance of each sentence/phrase with respect to overall meaning conveyed by the document(s). Therefore, the evaluated semantic similarity of each sentence and title and/or query is explicitly taken into account.

3.3 Sentence-sentence semantic Similarity:
To determine the similarity between two sentences, say, a and b, consisting of the sets of terms A and B, each term in A and B is first tagged with their POS (part of speech). It is then determined which noun each adjective describes and which verb each adverb describes. This is done by attempting to find the closest noun or verb following the adjective or adverb, and if none are found the closest noun or adjective preceding the adjective or adverb is used. The adjective and adverb lists are also expanded with exact synonyms from WordNet. The linguistic quantifiers, which may indicate the relative importance of a term within a text, are also associated with nouns in the same way. Typically, linguistic quantifiers are determiners which express information about relative or absolute quantity. The list of linguistic quantifiers is based on Bond's list [12]. In this study, one limited to two classes of linguistic quantifiers: those which induce an increasing order of relevancy like "very", "more", and those inducing a decreasing order like "less", "none", etc.
The similarity score for the sentence is therefore calculated by finding an average of the score for each noun or verb in both of the sentences. First the set of nouns and verbs for each sentence must be found.

$$NVA = \{x \in A : POS(A) = noun \cup verb\}$$

$$NVB = \{x \in B : POS(B) = noun \cup verb\}$$

Next, the best weighted match for each noun and each verb in both sentences must be determined.

$$BA = \{(u,v) : k_{uv} match(u,v) = \max_{i \in NVB}[k_{ui} match(u,i)] \ and \ u \in NVA\}$$

$$BA = \{(u,v) : k_{uv} match(u,v) = \max_{j \in NVA}[k_{jv} match(j,v)] \ and \ v \in NVB\}$$

Where

$$k_{ij} = \begin{cases} 2 \ \ if \ both \ i \ and \ j \ have \ an \ increase \ quantifier \\ 2 \ \ if \ both \ i \ and \ j \ have \ an \ decrease \ quantifier \\ 0.5 \ \ if \ i \ and \ j \ have \ opposit \ quantifier \\ 1 \ \ otherwise \end{cases}$$

And

$$match(u,v) = \begin{cases} (sim(u,v)+1)/2 \quad \text{if matching adjectives or adverbs were found} \\ \text{sim(u,v), otherwise} \end{cases}$$

where sim(u,v) is determined using either Jiang and Conrath's [6] semantic similarity $Sim_{JC}(u,v)$ or Lin's similarity [7] measure $Sim_L(u,v)$.
Finally the semantic similarity between sentence a and b is calculated by average the nouns and verbs semantic similarity as

$$SimSem(a,b) = \frac{\sum_{(u,v)\in BA} \mathrm{k}_{uv} \, match(u,v) + \sum_{(u,v)\in BB} \mathrm{k}_{uv} \, match(u,v)}{\sum_{(u,v)\in BA} \mathrm{k}_{uv} + \sum_{(u,v)\in BB} \mathrm{k}_{uv}}$$

The effect of this is to get a score which depends on every noun and verb in both sentences. In cases where a matching pair of adjectives or adverbs is found the score will be increased but not exceeding 1. If linguistic quantifiers are found, these are used to weight the average. The word-pair is weighted more highly where matching quantifiers are found and the weighting is reduced when opposite quantifiers are found. The above expression is also used to determine the score attached to the semantic similarity of the sentence to the query and the title, if any.

Using the abovementioned features, we are able to give a score to each sentence in all documents signifying their importance. The next section describes how the scoring takes place.

3.4 Scoring the Sentences:
The score for each sentence (score(i)), is generated based on the linear combination of the weighted features computed as described in the previous steps. The formula used for scoring each sentence is:

$$\text{Score(i)}=\frac{(\alpha Sim(s_i,\mathrm{T})+\beta Sim(s_i,Q)) \ n(s_i) \ (F_{NE}(s_i)+1) \ P(s_i)}{N(NE+1)}$$

Where:
- N is the total number of sentences in the document

- $n(s_i)$ is the number of sentences that have semantic similarity score bigger than a pre-defined threshold value
- $P(s_i)$ is the sentence position weight. For simplicity.
- $Sim(s_i, T)$ and $Sim(s_i, Q)$ are for the Semantic Similarity between the Title and the Query, respectively, and the sentence (i) determined using the sentence-sentence semantic similarity previously described.
- NE is the number of Named Entities in the document
- $F_{NE}(s_i)$ is the number of Named Entities contained in the sentence (i)

The rationale behind the preceding is to allow the score assigned to the sentence si very much dependent on the evaluation of the semantic similarity of si to both the title and the query using a convex combination of both entities. This output is weighted by n(si), which expresses, at some extent, the frequency of the sentences in the document(s) that are semantically similar to si up to some threshold μ, as well as the number of Named Entities in the sentence and its position. The positioning parameter is motivated by the observation that usually, beginning and end of the document contains more information regarding the context of the underlying document (s) as authors attempt to provide concise overview at the beginning and concluding remarks at the end. But, obviously this is very much context dependent. The weighting parameters α and β ($\alpha + \beta = 1$) are left open to the choice of the user depending on his/her prior knowledge about the relevance of the title and/or query. In the absence of any further evidence, the default values are 0.5 each, which is in agreement with the principle of insufficient reason in statistics.

3.5 Generating Summaries:

A summary is generated by choosing the most important sentences in a document (or the highest scoring) and arranging them in chronological order to insure the readability of the generated summary. Multi-document summaries are generated in a similar fashion by computing sentences scores in each document separately and then choosing the highest scoring sentences from all documents to generate multi-document summaries.

Handling the information redundancy between sentences and within each sentence was not completed in time and thus was not part of the system we used to participate in TAC 2008.

4. Evaluation

To evaluate our system, we participated in TAC 2008 for the first time even though some major components were not fully implemented in our system yet (i.e. redundancy checking and linguistic qualifiers handling). Next, we present results obtained from the automatic evaluation performed by NIST using ROUGE [8] and BE [9] metrics, and the manual responsiveness measure.

4.1 Test Data and Metrics:

For the TAC 2008 update task, we adopted the Jiang & Conrath [6] method when computing the semantic similarity between words. The redundancy handling component was not completed in time and thus the system used when participating in TAC 2008 did not handle sentences redundancy.

The provided test dataset comprised 48 topics. Each topic had a topic statement and 20 relevant documents which had been divided equally into 2 sets: A and B. The set A always chronologically precedes the documents in set B. The provided test dataset was taken from the AQUAINT-2 collection of news articles.[2]

All of the submitted summaries were truncated to 100 words. NIST conducted manual evaluation of summary content based on the Pyramid Method. Four different NIST assessors would create 100-word model summaries for each document set that addresses the information need expressed in the topic statement.

Each participant team was requested to submit up to 3 runs ranked by priority (1-3). Our team submitted two runs: one (run # 1) has more weight given to the topic statement, and the other (run # 34) has more weight given to the headlines. In the abovementioned scoring formula, α was given a value of 0.75 for run 1, and 0.25 for run 34.

4.2 Results:

In the update task of TAC 2008, 57 peer summaries were manually evaluated with the pyramid method, and 71 were evaluated using ROUGE and the Basic Elements evaluation package [9].

Table 1 shows the average Recall, Precession and F-measure for the Rouge1, Rouge2, and RougeSU4 evaluations on the two runs we submitted. It can be noted that in both runs, the system generally ranked

[2] See http://www.nist.gov/tac/tracks/2008/summarization/update.summ.08.guidelines.html for more details about the Task.

higher in Recall than Precession. This suggests that the system is better at finding relevant content than it is at removing irrelevant content. Also, it can be noted that the run which more weight given to the topic statement generally achieved better ROUGE scores than the other run with more weight given to the headlines.

	Run 1			Run 34		
ROUGE	**Avg R**	**Avg P**	**Avg. F**	**Avg R**	**Avg P**	**Avg F**
1	0.34463	0.33866	0.34148	0.34022	0.33372	0.33680
2	0.08091	0.07933	0.08008	0.08080	0.07912	0.07991
SU4	0.11852	0.11634	0.11737	0.11706	0.11471	0.11583

Table1: The Rouge Scores obtained by our system in the two runs we submitted.

Table 2 shows the automated evaluations average scores obtained by our submitted runs (with their ranks) in comparison with the 71 peer summaries submitted by the rest of the participants.

Evaluation	Run (1)	Run (34)	Best	Worst
ROUGE2-R	0.08091 (25/71)	0.08080 (26/71)	0.10382	0.03343
ROUGESU4-R	0.11858 (23/71)	0.11713 (29/71)	0.13646	0.06517
BE	0.04964 (24/71)	0.04903 (28/71)	0.06462	0.01337

Table2: the automated scores (and ranks) obtained by our system in comparison with the rest.

The evaluation in TAC2008 included human judgments of linguistic quality. Table 3 shows the results and the rank of our system in respect with the rest in the manual evaluation. The metrics shown in the table are: responsiveness which is how well the summary addresses the user's information need; and linguistic quality. The linguistic quality score is guided by consideration of the following factors:

1. Grammaticality
2. Non-redundancy
3. Referential clarity
4. Focus
5. Structure and Coherence

with scores between 1 (very poor) and 5 (very good).

	Run (1)	Run (34)	Best	Worst
Avg Linguistic Quality	2.719 (12/58)	2.76 (11/58)	3.073	1.312
Overall Responsiveness	2.427 (15/57)	2.385 (18/57)	2.667	1.198

Table 3 : Manual Evaluation Results

5. Future Work

We plan to add and improve many aspects of the system we developed, especially in the post-processing part. Among the ideas we plan to integrate are the following:

- Implement redundancy checking and remove repeated information. We think that implementing this feature will greatly enhance the evaluation results. This should be done from two different perspectives: First, removing repeated or non-essential content from within sentences such as relative clauses (which can be done in the last stage just before choosing the summary highest scoring sentences by adding a new metric: redundancy penalty affecting the repeated sentences score). Second, relating the chosen summary sentences with each other and trying to maximize the information content diversity between sentences to achieve the highest possible comprehensiveness in the generated summary. To achieve the later, the semantic similarity between the summary candidate sentences can be checked against a previously set threshold and thus reducing the score for those sentences containing repeated data.
- Try to find a method to automatically optimize the weight of the dynamic features. Currently, the weights are assigned manually based on the user's observations. Implementing this will require great deal of analysis to the syntax of the text in each sentence, which has not been deeply explored in our system.
- Compressing the summary sentences to allow for more information to be presented in the summary at the same or shorter length. Syntactic trimming which has been studied in previous work [3] is what we are currently exploring and hoping to improve and implement in our system.
- Meeting the word limit in our system was achieved by simply iterating through all the highest scoring sentences to replace the last summary sentence with the next shorter and high scoring sentence. This means that in some cases, none of the sentences are chosen (in the case when replacing the last summary sentence with any other will yield a summary longer than the required word-limit) and

thus sentences with valuable and relevant content to the user query are not added because of their length. This will need further investigation and can be partially overcome by generating shorter forms of long sentences (compressing the summary sentences) and eliminating non-important sentences before the processing stage using "shallow parsing " techniques similar to [10].

- We plan to use co reference resolution to enhance the quality of our generated summaries. For example, some sentences might contain references to important entities such as "President Bush" in the form of one word "he". We think that replacing the pronoun with the Named Entities before processing the summaries should give better scores for our summaries [11].

6. Conclusion

In this paper we present our on-going work on building a query-focused multi-document summarization system and the evaluation results for the system in the update task of TAC 2008. The results suggest that our overall system rank can be placed in the middle tier when compared with all the participants in the task for this year. In future work, we plan to apply and experiment with more detailed measures to handle different aspects such as redundancy, comprehensiveness and length, and automatic weight optimization for the dynamic features.

References

1. Radev, D., et al., *MEAD ReDUCs: Michigan at DUC 2003.* Proceedings of DUC 2003, 2003.

2. Radev, D.R. and G. Erkan, *The University of Michigan at duc2004.* Proceedings of Document Understanding Conference Workshop, 2004: p. 120-127.

3. Zajic, D., *Multiple Alternative Sentence Compressions as a Tool for Automatic Summarization Tasks*, in *Department of Computer Science*. 2007, University of Maryland.

4. GATE. *GATE - General Architecture for Text Engineering*. 2007; Available from: www.gate.ac.uk.

5. Sekine, S. and C. Nobata, *Sentence Extraction with Information Extraction Techniques.* Workshop on Text Summarization 2001, 2001.

6. Jiang, J. and D. Conrath, *Semantic Similarity based on Corpus Statistics and Lexical Taxonomy.* Proceedings of International Conference Research on Computational Linguistics,Taiwan, 1997.

7. Lin, D., *An information-theoretic definition of similarity.* Proc. 15th International Conference on Machine Learning, 1998: p. 296-304.

8. Lin, C.-Y., *ROUGE: a Package for Automatic Evaluation of Summaries.* Proceedings of the Workshop on Text Summarization Branches Out (2004), 2004.

9. Hovy, E., C. Lin, and L. Zhou, *Evaluating DUC 2005 using Basic Elements.* Proceedings of the HLT/EMNLP Workshop on Text Summarization DUC 2005, 2005.

10. Dunlavy, D., et al., *Performance of a Three-Stage System for Multi-Document Summarization.* 2003.

11. Conroy, J.M., et al., *Left-Brain/Right-Brain Multi-Document Summarization.* Document Understanding Conference Workshop at HLT/NAACL 2004, 2004.

12. F. Bond, Determiners and Number in English contrasted with Japanese, as exemplified in Machine Translation, University of Queensland, 2001