INVESTIGATING THE EXTRACTIVE SUMMARIZATION OF LITERARY NOVELS

Hakan Ceylan

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

December 2011

APPROVED:

Rada Mihalcea, Major Professor
Deniz Yüret, Committee Member
Kathleen Swigger, Committee Member
Paul Tarau, Committee Member
Barrett Bryant, Chair of the Department of Computer Science and Engineering
Costas Tsatsoulis, Dean of the College of Engineering
James D. Meernik, Acting Dean of the Toulouse Graduate School

Ceylan, Hakan. Investigating the Extractive Summarization of Literary Novels. Doctor of Philosophy (Computer Science), December 2011, 112 pp., references, 109 titles.

Due to the vast amount of information we are faced with, summarization has become a critical necessity of everyday human life. Given that a large fraction of the electronic documents available online and elsewhere consist of short texts such as Web pages, news articles, scientific reports, and others, the focus of natural language processing techniques to date has been on the automation of methods targeting short documents. We are witnessing however a change: an increasingly larger number of books become available in electronic format. This means that the need for language processing techniques able to handle very large documents such as books is becoming increasingly important.

This thesis addresses the problem of summarization of novels, which are long and complex literary narratives. While there is a significant body of research that has been carried out on the task of automatic text summarization, most of this work has been concerned with the summarization of short documents, with a particular focus on news stories. However, novels are different in both length and genre, and consequently different summarization techniques are required. This thesis attempts to close this gap by analyzing a new domain for summarization, and by building unsupervised and supervised systems that effectively take into account the properties of long documents, and outperform the traditional extractive summarization systems typically addressing news genre.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Dr. Rada Mihalcea, for her continuous support, and being patient with me during my PhD study at the University of North Texas. Without her guidance, it would be impossible for me to have the motivation that helped me find my way through the many difficulties I have faced through the course of my doctorate education.

Secondly, I would like to thank Dr. Kathleen Swigger. Apart from being my MSc advisor and one of my PhD committee members, Dr. Swigger has been a mentor, and a friend to me, and she has always been available no matter what problems I have had. Her help has been instrumental in my success at graduate school.

Many thanks to Dr. Deniz Yuret for being in my committee, and for his willingness to spend time and share valuable comments even though he was oceans apart. His research work has always been truly inspirational to me. I also would like to thank my committee member Dr. Paul Tarau for spending the time and effort to provide valuable comments, and criticism.

My special thanks go to family. My mother, Nuriye Ceylan, my father, Hami Ceylan, and my sister, Bilge Ceylan, showed endless support and undying love during the entire time I have spent away from them in graduate school. This thesis would not have been possible without them.

Finally, I would like to thank my friends in Language and Information Technologies Laboratory for going through every good and bad moments together with me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Motivation and Problem Definition

Due to the vast amount of information we are faced with, summarization has become a critical necessity of everyday human life. When we apply for a job, we first prepare a resume, which summarizes all of our achievements to date to a few pages. In order to select the relevant papers to read, the first thing the researchers look at is the abstracts, which summarize the work and the contributions of the authors. The CNN's online web page provides 3-4 bullets for each article, which points out to the main events of the story. People prefer to watch the trailer of a movie first before they decide to see it, as the movie trailers give a glimpse of the plot, and a taste of what is to come. Many other examples can be added to this list, including book reviews, web page snippets given by search engines, newspaper headlines, and even computer programs, which can be seen as a compact and precise implementation of an idea. In general, as stated in [57], the goal of summarization is to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's needs.

It is important to note that, in the above examples, the notion of a summary takes a different form depending on the information source and the application. In this thesis, however, documents are the only source of information. Furthermore, the documents are assumed to contain only textual material, thus in a general sense, they can be thought of as a means for the written representation of thoughts. Hence it is assumed that the documents do not contain other types of media such as images or videos. Therefore, throughout the thesis, the following definition of a summary is adopted:

DEFINITION 1.1. A summary is a text that is produced from one or more texts, that contains a significant portion of the information in the original text(s), and that is no longer than half of the original text(s) [81].

Note that the above definition suggests that there may be more than one source document involved in the process. This task, referred to as multi-document summarization, attempts to generate a single summary from a collection of different but conceptually related documents. Although advances in multi-document summarization is of great interest, the task described in this work is only concerned with single document summarization. Thus the term summarization refers to generating a summary from only one source document according to the definition above.

As in the case of machine translation and other fields of artificial intelligence, it is possible to build summarization systems that make use of human assistance to a degree. However, this thesis is only concerned about fully automatic summarization systems, which require no human assistance in generating the summaries.

Given that a large fraction of the electronic documents available online and elsewhere consist of short texts such as web pages, news articles, scientific reports, and others, the focus of natural language processing techniques to date has been on the automation of methods targeting short documents. We are witnessing however a change: an increasingly larger number of books become available in electronic format, in projects such as Gutenberg[1], Google Book Search[2], or the Million Books project[3]. Similarly, a large number of the books published in recent years are often available – for purchase or through libraries – in electronic format. This means that the need for language processing techniques able to handle very large documents such as books is becoming increasingly important.

This thesis addresses the problem of summarization of novels, which are long and complex literary narratives. While there is a significant body of research that has been

[1]http://www.gutenberg.org
[2]http://books.google.com
[3]http://www.archive.org/details/millionbooks

carried out on the task of automatic text summarization, most of this work has been concerned with the summarization of short documents, with a particular focus on news stories. However, novels are different in both length and genre, and consequently different summarization techniques are required. In fact, the straight-forward application of state-of-the-art summarization methods leads to poor results on this domain. This is not surprising since these systems were developed specifically for other genre types, which contain documents that are much shorter than novels.

Since a significant amount of interest and effort has been given to the summarization of short documents, there is a considerable amount of datasets, and tools developed by the research community for these domains. In contrast, with the only exception of [40, 41], in which the summarization of short fiction stories are investigated, no attempts were made for the summarization of long documents, hence neither a dataset collection nor a tool development is presented for this task. This thesis tries to close this gap by introducing a new dataset, benchmarks, and analysis methods, and by describing systems that make use of novel features as well as the existing ones for the summarization of a new domain, literary novels.

In order to be able to carry out various experiments and to draw conclusions on them, the first thing needed is the collection of a new dataset for this genre. To date, a set of novels consisting of 66 books, which are all literature classics, was collected together with their summaries, which are human-generated abstracts. Currently there are a total of 1109 abstracts in the dataset in three categories, where there are a total of 442 in the first category, 290 in the second one, and 377 in the third. The majority of the interest, however, is given to the summaries in the first category. The definitions of these cateogries are given in detail in Chapter 3.

However, due to the specific constraints required by each study, the dataset used in the experiments may differ across some of the chapters. Even though they are all drawn from the same collection, some of them are tailored for the specific application, for example

by eliminating the novels that do not meet certain requirements the application needs. Each such process is described as a separate section in the chapters where it is applied.

1.2. Research Questions

Perhaps the most obvious question that comes to mind prior to starting a research for a new domain in an existing field is whether this new genre exhibits certain characteristics that make it suitable for further investigation. Chapter 4 tries to answer this question by considering the applicability of the first stage of a text summarization model to literary novels. This objective is performed by analyzing the parameters of a dynamic programming algorithm, which was previously applied to news articles in order to align the abstracts to the source documents. Furthermore, the chapter makes a second contribution by distinguishing between the two abstract categories in the dataset for their suitability to the automatic summarization task.

Having analyzed the question of whether literary novels are suitable for perhaps the most critical stage of text summarization, i.e. extraction, the thesis seeks the answer to the question of how far one can go with this stage in various domains. As the next chapter explains in detail, the extraction step of the text summarization model assumes that certain units from the input source are copied and pasted verbatim to the generated summary. Given that there are a finite number of such units in any text, in theory, it is possible to compute the limits of this copy-paste step via a simple enumeration. However, the exponential complexity resulting from this combinatorial problem imposes a big challenge. Chapter 5 describes a novel attempt to overcome this challenge by using large computational resources as well as a divide-and-conquer approach that makes it possible to estimate the theoretical limits of this problem via probability density functions induced over an entire dataset. Another contribution of this chapter is the comparison results of application of this methodology to three other domains in addition to the literary domain that makes the foucs of this dissertation.

Chapter 6 describes the first summarization system in this thesis, which builds upon methods previously found successful in the newswire domain, but takes into account various

attributes of long documents, and develops novel features for them. The primary purpose of the research here is to confirm the earlier insights by implementing a real system, and evaluating it on the newly constructed dataset of literary novels. It is shown in this chapter that the existing methods prove insufficient on the new dataset, and even an ad-hoc combination of features that consider the length and the structure of the long documents give a big boost on the evaluaion scores.

The research question addressed in Chapter 7 is concerned with the combination of summarization systems in a supervised fashion using machine learning algorithms. The features used in this process are various unsupervised extractive summarization systems that are built on either existing ideas or new ones that are adjusted for the literary novels domain. Furthermore, the learning process also makes use of the exhaustive evaluation results in order to calculate the labels for the training instances. For comparison purposes, the process is applied also to a dataset from the news genre. This study confirms that the prevelant features differ for each domain, and also shows that a formal supervised approach to combining different extractive summarization systems outperform each individual system.

The final contribution of the thesis, given in Chapter 8, is to break away from the sentence extraction techniques, and look into the steps of bringing the summary sentences into more compact form. This work considers the interpretation phase of the text summarization model, which is also explained in the next chapter. The main purpose at this step is to improve the summarization performance by compressing the compressing each extracted sentence without losing too much information so that more sentences can be included in the summary. A novel sentence compression approach is used at this stage. The algorithm uses information theoretic measures and gives considerable success on a corpus developed as a testbed for sentence compression systems. Furthermore, the design of this algorithm resulted in another important contribution; a fast indexing and retrieval system for large n-gram corpora. This system is also described in detail in Chapter 8.

Hence the goal of this thesis is to investigate a new domain, literary novels, for extractive summarization. This goal is successfully achieved in many aspects; i.e. by analyzing

the applicability of extraction to literary novels, by quantifying the limits and success of this extraction, by building unsupervised and supervised extractive summarization systems, and by adapting an external sentence compressor to the system. It should be noted that the purpose here is not to build the best possible summarization system for this new domain, but rather to lay a new ground by constructing novel frameworks in all the mentioned aspects, and thus providing a proof of concept.

While the thesis builds a framework for many aspects of summarization, it also leaves out some of the important ones such as sentence planning, and sentence fusion. These two aspects are parts of the interpretation, and summary generation steps (see Chapter 2), and are used to bring the summaries into more coherent and human-readable forms. Since this thesis is concerned with extractive summarization, those aspects are out of its scope.

As a final note this thesis benefits from publications [67, 10, 12, 11] in the following conferences; EMNLP 2007, CICLING 2009, NAACL 2010, and ACL 2011. Furthermore, the work in chapter 7 is also to be submitted to an upcoming natural language processing conference.

CHAPTER 2

EXTRACTIVE SUMMARIZATION

2.1. Overview of Text Summarization

There are two main points of summarization that usually separates it from other tasks such as text compression, information extraction, question answering, etc. These two points are the summary text being in a condensed form of the original information source (i.e. the document), and being produced for human consumption i.e. to satisfy the user's needs. Note that these two properties, which do not necessarily exist together in other fields, are the main goals of the field of summarization. It should, however, be noted that summarization can benefit from other fields. For example, a summarization system focused on providing summaries based on user queries may very well benefit from question answering, although in general it is a much broader field.

The restrictions that are employed in Section 1.1 in order to define the focus of this thesis, represent an instance of a broader classification of summarization systems, which is given in detail in [92], where three main classes that affect the summarization systems are identified. These classes consist of the input factors, purpose factors, and the output factors.

The input factors define the characteristics of the information source, such as its language, length, structure, and genre. For example a news article differs in its input factors than an article found in a medical journal. Due to the differences in genre, the discourse structure of the texts in both information sources vary as well as their length, and the linguistic style. Therefore the summarization systems designed for these domains typically take into account these differences.

The purpose factors define the use of summaries from the perspective of the intended audience, as well as the application. Factors such as time, location, and formality may

also be included. Continuing with the same example, summaries of news articles are generally intended for ordinary users, whereas the abstracts in medical journals are intended for researchers in the medical field. The web page snipptes given by the search engines are also produced for ordinary users, but specific to a query, which in this sense examplify the difference from the application point of view.

Finally, the output factors are concerned with the style, and format of the summary as well as the coverage and the compression aspects. Note that the output factors are affected by both the input and the purpose factors in the sense that the style of the information source and the intended audience usually constrain the output style of the summary. However these factors alone do not determine the output factors as a whole. The most widely considered output factors are coverage, compression, style and form.

The coverage refers to parts of the information source mentioned in the summary. A summary can have either comprehensive or selective coverage. For example; the systems developed for query-focused summarization employ selective coverage as they are only concerned with the parts of the information source related to a query. On the other hand, generic summarization systems attempt to find all the salient points of the information source, and produce their concise representation, thereby giving comprehensive coverage.

The compression factor defines the length of the summary compared to its information source, and is usually given as a ratio. Thus a compression ratio of 80% suggests that the length of the summary is 20% of the length of the input text. Here, the length of a text document is usually measured in terms of the number of words, and rarely the number of sentences. The compression ratio is mostly dependent on the genre of the input document and the intended application.

Regarding the style of the summaries, the literature typically distinguishes between indicative vs. informative summaries [8, 35, 93, 57, 92]. The indicative summaries help the readers assess the aboutness of the content of the information source so that the reader may decide whether to do further reading on it. Indicative summaries do not elaborate on the salient points of the text but rather just point to them without giving any details. Hence

they usually provide information about the purpose and general scope of the text without mentioning the specific events, results or conclusions. In contrast, the informative summaries provide the most important points of the text in the summary, and attempt to cover as many of these aspects as possible.

The form (or derivation) of the summary is usually given as one of the two types; extracts, concerned with the identification of the information that is important in the input text, and abstracts, which involves a generation step to add fluency to a previously compressed text [35, 93, 57, 92]. The extracts reproduce parts of the information source, which can be a single word, sentence or an entire paragraph, verbatim in the summary. Moreover the extracted parts need not be contiguous so the summaries are not usually coherent in extractive summarization. The abstracts, however, produce newly generated text usually with novel phrases or sentences that are not found in the original text. In contrast to extracts, the abstracts are coherent, and may deliver the same amount of information with a shorter summary as they may benefit from paraphrasing, and fusion of the ideas in the information source.

In the automatic summarization literature, abstracts are generally considered as the further processed, or the transformed form of the extracts. This view results from the widely accepted three-phase model [93, 59, 37, 49, 57, 92]. The model consists of the topic identification, interpretation, and the summary generation steps. Although the phases will be referred as such in this thesis, it should be noted that there is not a consensus among the authors on these names. For example, in [93, 92], the first two phases are given as interpretation, and transformation. Instead, [57] uses the names analysis, transformation, and synthesis. Perhaps this unagreement results from the fact that these phases are not clear-cut but are rather intervened with each other. Moreover, [69] defines four phases rather than three, which puts an extra reinterpretation step as the third.

Topic identification is the first stage of the three-phase model of automatic text summarization, and its goal is to analyze the source text, represent it internally, and identify the

most important units in the document, i.e., phrases, sentences, or paragraphs. The extractive summarization systems only employ this stage, and simply output a ranked list of the units according to a compression ratio criterion. In general, for most systems, sentences are the preferred units in this stage, as they are the smallest grammatical units that can express a statement. Scoring of the sentences are generally done using a module that combines the scores of independent modules, either in a supervised or unsupervised fashion.

In the interpretation step, the units of text identified in the previous stage are fused and put into a new representation. Hence, this stage is also referred to as concept fusion. The topics found in the identification stage are aggregated and expressed in novel ways. Thus, this stage distinguishes the systems geared towards producing abstracts from the extractive summarization systems. The idea of aggregating the identified topics, and formulating them in new ways gives these systems a greater chance of condensing the source content. This intuitive notion is supported by the study in [61] in which it was found for 10 news articles that the extracts are 2.76 times longer on average than the corresponding abstract which covers the same points of the source document. Therefore an abstract-type summary usually carries more information compared to an extract-type summary of the same length.

Note that in order for the summarization systems to perform the interpratation phase, an external knowledge is required as this stage involves making inferences so that the sentences can be rephrased or rewritten. For example, a phrase such as "He bought tulips, roses, and daises" can be brought into a more compact form "He bought flowers". Therefore the knowledge acquisition is a major obstacle for summarization systems to perform the interpretation stage. Hence the systems to date mostly concentrated on the extraction step, and very little has been done on the second phase [48, 65, 84, 83].

The third stage of the three-phase model is summary generation, and it is used to bring the internally represented summary to a more coherent and human-readeble form. Natural Language Generation techniques such as content planning are used in this step in order to avoid contradictory information, and repetition of sentences, as well as resolving coreferences, and adding pronominalizations, and discourse cues [58, 5]. Another technique

related to this stage but also studied independently is sentence compression, which is the task of producing a shortened version of a sentence by retaining the salient information while keeping the grammatical constraints intact. Several successful sentence compression techniques are developed to date [43, 14, 15].

Insights for the field of automatic summarization are obtained by investigating the way professional human abstractors compose their summaries [23, 69, 17]. Note that the professional abstractors mentioned here are not experts on the fields in which the source articles are written. An empirical study of six human abstractors, given in [23], found that the process followed by humans can be broken down into three steps; document exploration, relevance assesment, and summary production. In document exploration, the summarizer gets familiar with the input features of the document, such as style, format, and organizational layout. In relevance assesment, the theme of the document is identified, and the salient passages and sentences are marked. Finally, in the summary production step, the abstractors "cut-and-paste" the previously identified passages into the summary by reorganizing and bringing them into a coherent form. Furthermore, the study suggests that the human abstractors never read the entire document but rather skim over it as the whole summarization process is completed in about 15-20 minutes. Finally, the most important features the abstractors use are discovered to be the cue phrases, location of the sentences, and title headings.

2.2. Overview of the Previous Work

Various approaches to automatic summarization have been considered to date, in which most of the efforts have been on the topic identification phase. The earliest two studies cited heavily in the literature are [55], and [6]. In [55], the term frequency feature is used along with stemming, and stop words removal. In [6], the sentence position feature is analyzed and it was found that in 85% of the 200 paragraphs considered, the topic sentence came first, and in only 7% it was found to be the last sentence. These two features, with the addition of cue words, and title headings, are used later in [22], in which the term frequency feature is also weighted with dispersion, i.e. inverse document frequency. The weighted

features are then combined using a linear model, and a manual evaluation found 44% of the generated summaries successful.

With the advancement of statistical techniques in natural language processing, as well as the machine learning methods, starting with 1990s, the field of automatic summarization received a considerabe attention from the natural language processing and information retrieval communities. Moreover, starting in 2001, Document Understanding Conferences (DUC)[1] are organized by the National Institute of Standards and Technology (NIST), which presented several summarization tasks for the systems to participate. Among the many competitions carried out, some of the tasks were; the generation of short (10 words) summaries, exploratory summaries, multi-document summaries, and event or question focused summaries. The task of 100 words single-document summarization of news articles was only employed in 2001, and 2002, as none of the systems participated in these years were able to significantly outperform the lead baseline, which extracts the first N sentences of the article until the length limit is reached. Hence this task is removed in the upcoming years.[2]

One of the first supervised systems in text summarization, which uses a naive Bayes classifier, is described in [44], where each sentence in the source document goes through a binary classification, i.e. whether to be included in the summary or not. The features used in this classifier were mainly the ones in [22] with a few new ones such as the length of the sentence. Based on a manual matching of the medical abstracts to source text, the study found the most successful system as the one that is using position, cue phrases, and sentence length features together. In [97], the same classifier is used on a different data, which consisted of scientific articles for which the summaries are written by the authors of the articles themselves, rather than the professional abstractors. Another study that extends the idea of naive Bayes classification is given in [1] which considered richer features such as word collocations. Furthermore, an attempt is also made in [1] to bring some coherence into

[1] http://duc.nist.gov/

[2] Starting 2008, the scope of the conference is extended by including new tracks such as textual entailment recognition, and knowledge base population, and the name of the conference is changed to Text Analysis Conference (TAC) (http://www.nist.gov/tac/).

the summaries by performing a shallow discourse analysis of the source text, and by using coreference resolution.

Instead of a naive Bayes classifier, a decision tree is used in [49], which combined even more features. The features used in the study ranged from very simple ones such as whether a sentence has proper names, or numerical data, to more complex ones such as measures of lexical connectivity. However, even these more complex features were still based on the shallow statistical techniques. An important finding of the study was that even a simple ad-hoc combination of the features did as well as the decision tree classifier most of the times, and sometimes even exceeded it. Another important aspect of this study was disregarding the assumption of feature independence. In a later work, this notion is also considered via a maximum entropy model [75].

It should be noted that the behaviour of each feature would vary across domains as the discourse structure of the documents would be different for each domain. For example, the position of the sentence in the source document is found to be the most important feature for the newswire domain [73]. However, as it is shown in Chapter 6 of this work, the leading sentences in literary novels yield to poor performances. Therefore some of the features may need tuning for each domain. The work of [51] describes a study that attempts to optimize the positional feature across different genres. The study is based on developing an optimal position policy (OPP) by ranking sentences for the highest yield, which is measured againist the topic keywords given for each article of the Ziff-Davis Corpus.

The work of [89] describes a fast query based multi-document summarization system that only uses simple term frequency based features. The learning setting presented is similar to the work described in Chapter 7 except that the training instances are labeled using the word overlap between the sentence and the model summaries instead of exhaustive evaluations used here. A more complicated framework that takes the interdependencies between the sentences into account using conditional random fields is presented in [90]. In [105], an attempt is made to learn to predict the appearence of individual terms in the references, independent from the sentence selection procedure. Finally, the work in [46]

presents a system that jointly learns to optimize diversity, coverage, and balance. Hence the proposed system seeks to minimize redundancy by including the main points of the summary from as many aspects as possible.

Some of the more statistically enriched approaches that consider information-theoretic features are described in [70, 53]. In [70], information gain is used to weight the terms of the sentences. As usual the sentences which have the highest term weights in total are extracted. In [53], the sentence scores are determined based on the number and the length of noun phrases, which indicate the informativeness of a sentence based on the linguistic theory of code quantity principle. Another type of a novel statistical feature based on eigenvector centrality is given in [68, 24, 66]. The system described in [68] ranks among the top systems on DUC 2001, and DUC 2002 datasets. Another top performing system in DUC competitions was MEAD [82], which used the concept of centroids, indivudual words or phrases associated with an importance score, which is based on the idea of tf.idf [87].

Systems that go beyond using shallow approaches by performing semantic processing, exploiting the document structure are described in [65, 3, 60, 62, 4]. In [65], the SUMMONS system is described, which uses predefined templates, which are limited to a specific domain, to identify the sentences to include in the summary, and a linguistic component that makes use of symbolic techniques to determine the structure of the output sentences, and to present the identified sentences in a refined form. The linguistic component presented in the study consists of three parts; the ontologizer, lexical chooser, and the sentence generator. In [3], a new algorithm to compute the lexical chains is introduced and used for the topic identification stage. The system developed segments the text first based on a linear discourse structure assumption, identifies lexical chains in each segment, and finally extracts sentences based on the strongest chains found. The identification of lexical chains made use of external knowledge sources such as WordNet [26], as well as part of speech taggers, and shallow parsers. A break away from the assumption of linear discourse structure is given in [60, 62], where the rhetorical structure theory (RST) is used to develop a parser which produces the discourse tree of the text, which represents relations between text units. Then,

a summarization algorithm that extracts sentences based on the partial ordering between these textual units is given. Finally, an approach that considers the text structure, but this time in terms of content, is described in [4], where hidden Markov models (HMM) are used to learn a content model of a specific domain from an unannotated corpus of documents. A content model is basically used to assign topic labels to sentences in the source text with associated probabilities. A summarization system based on these models first finds the most likely topic in the source text that should be included in the summary, and then selects the top sentences with the same topic based on their probabilities.

Regarding the use of external knowledge for the purpose of extraction, a novel work is given in [95], in which the task was to mimic the summary generation of an online CNN article, which is given in 3-4 bullets, where each bullet is a human-generated sentence describing the highlights of the story. In addition to using a neural network classifier, the novel contributions of the work was integrating two external knowledge sources to the system; search query logs and Wikipedia entities about the story. The resulting classifier significantly outperforms the lead baseline, in 70% of the dataset. However, during the evaluation, no attention was paid to keep the length of the system and the baseline summaries equal in terms of the number of words. In other words, no chopping is done in the sentence level.

Another important aspect of automatic summarization is the evaluation of summaries. According to [57], the task should include metrics such as coherence, conciseness, grammaticality, readability, and content. The manual evaluations performed in the DUC/TAC competitions tackled some of these aspects by using two metrics; content responsiveness, the degree to which a summary responds to the information need contained in the topic statement, and readibility, which measures the linguistic aspects such as coherence, sentence structure, and grammaticality independent of the content. The human judges evaluated both the responsiveness, and the readibility using a five-point scale, from "very good" to "very poor." Another approach to evaluate the content of the summaries manually, also used in DUC, is the pyramid method, and described in [74]. In this method, the human assessors identify the informative units, which are words or phrases, in multiple human generated

summaries, and then weight them based on the number of occurences across the multiple summaries. Then a system generated summary is assigned a score based on these weights.

Even though manual evaluation is still the gold standard, it has many drawbacks. First, manual evaluation of summaries performed by humans is very time-consuming, as it is stated in [76, 50], the manual evaluation of summaries in the DUC competitions would require more than 3,000 hours of human work. Furthermore, it was also found that there is little agreement between humans for this task. This is perhaps understandable as there is no single unique or perfect summary for a source text. In addition, each summarizer might capture different aspects of the source text, and hence produce different summaries. This also suggests that the human summarizers might not be objective at all times. Therefore, given these facts, there is a need to automate this task in order to make fast, reliable, and unbiased evaluations.

One of the first automatic evaluation systems was proposed in [85], in which three methods were described in order to measure the content overlap between the reference (human) and the candidate (system) summaries. These three methods were cosine similarity, unit overlap, and the longest common subsequence. The study however did not consider how well these measures correlate with human judgments. In contrast, inspired from the BLEU automatic evaluation system applied successfully to machine translation, the study in [50] introduced a new system, ROUGE[3], which uses the same features as well as n-gram co-occurence statistics, and was shown to correlate well with human judgements. Although ROUGE gained huge popularity from the summarization community, it also received a lot of criticism due to the fact that it only used shallow n-gram matchings to judge the content of the summaries without considering the semantic structure or taking aspects such as cohesion and linguistic quality into account. Another system, Basic Elements (BE), described in [36] employs a linguistic analysis to a degree. The basic elements in a text are composed of the heads of the major syntactic constituents, and the dependency relations of their modifiers. Hence, these basic elements are extracted from the candidate and the reference summaries

[3]ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation

after parsing the texts with a syntactic dependency parser, and then compared against each other. This methodology is shown to have better correlations with humans than ROUGE, although the results were not statistically significant. Finally, very recently, a slightly more successful system than BE, DEPEVAL(summ) is introduced in [77]. The difference between BE, and DEPEVAL(summ) comes from the fact that while the former uses Minipar parser to extract the dependencies, the latter accomplishes this through Lexical-Functional Grammar dependencies.

CHAPTER 3

RESOURCES

3.1. Dataset

A first challenge encountered on the task of summarization of literary novels was the lack of a suitable data set, designed specifically for the evaluation of summaries of long documents. Unlike the summarization of short documents, which benefits from the data sets made available through the annual Document Understanding Conference (DUC) evaluations, there are not any publicly available data sets that can be used for the evaluation of methods for novel summarization.

The lack of such data sets is perhaps not surprising since even for humans the summarization of very long documents such as literary novels is more difficult and time consuming than the summarization of short news documents. Moreover, even though more and more books are becoming available in electronic format these days, they are typically protected by copyright laws that do not allow their reproduction, which consequently prohibits their public distribution.

The construction of the dataset used in this study started from the observation that several English and literature courses make use of literary novels that are sometimes available in the form of abstracts – to provide study materials, and ease access to the content of the books. Several publishers are identified that make these abstracts available online for books studied in the U.S. high-school and college systems, such as GradeSaver (http://www.gradesaver.com), SparkNotes (http://www.sparknotes.com), and CliffsNotes (http://www.cliffsnotes.com/). Fortunately, many of these novels are classics that are already in the public domain, and thus for most of them it was possible to find the online electronic version of the books on sites such as Gutenberg or Online Literature (http://www.online-literature.com).

Most of these novels are divided into chapters by their corresponding authors at the time of their writing. Furthermore, most of the summary publishers also follow the same format, and present a summary per chapter[1]. Hence the chapter information is clearly identified in the dataset if it is available.

In general, three different kinds of summaries are provided by the publishers mentioned above, each summary serving for a different purpose. The types of these summaries are named as objective summaries, interpretative summaries, and synopsis. The objective summaries are the ones that describe the plot of the novel, without any interpretation from the summary writer. The second type consists of analysis summaries that describe, in a subjective manner, the interpretation of the facts and of the main story in the novel. The publishers often give these summaries under Notes/Analysis/Interpretation sections, hence these summaries are referred to as interpretative summaries. Objective and interpretative summaries are collected in chapter level for each novel. The last type of summaries, synopsis, are similar in essence to objective summaries but much shorter. These summaries also describe the main story of the novel in an objective manner, but they do not come in chapter level. Hence these summaries are not as detailed as objective summaries but rather give a very high level description of the events in the novel. Synopsis are no longer than one or two pages.

It should be noted that the distiction between the objective and the interpretative summaries is different from the distinction between the indicative and the informative summaries. Recalling from Chapter 2 that, the indicative summaries provide the general idea of the text to the reader without getting into details, whereas the informative summaries try to provide all the important points. In this case, both the objective and the interpretative summaries are informative summaries as they are both concerned with the salient points of the story, although with a different style. The interpretative summaries can be classified as critical summaries, which are also distinguished by some authors as a summary style in addition to the informative and indicative summaries [57, 45].

[1]Sometimes a summary is given for a combination of some of the chapters.

OBJECTIVE SUMMARY: As the novel opens, the soldiers of a regiment are waiting for battle. After one of the men, a tall soldier, suggests that a battle is imminent, other soldiers argue against the notion. One of the young soldiers, Henry, a private, returns to the hut where the regiment is camped and thinks about war. He recalls his desire to enlist in the army, his mother's refusal to support the idea, and his eventual decision to enlist over her objections.

INTERPRETATIVE SUMMARY: The overriding impression of this first chapter is one of conflict. The Union soldiers await a physical battle with the Confederate troops in the area. The eminent external conflict is paralleled by the fight raging in Henry's mind. As the book opens, the reader sees the main character, a soldier waiting for his first battle, ironically engaged in an internal conflict with his own thoughts.

SYNOPSYS: The Red Badge of Courage is the story of Henry Fleming, a teenager who enlists with the Union Army in the hopes of fulfilling his dreams of glory. Shortly after enlisting, the reality of his decision sets in. He experiences tedious waiting, not immediate glory. The more he waits for battle, the more doubt and fear creep into his mind. When he finally engages in his first battle, he blindly fires into the battle haze, never seeing his enemy. As the next enemy assault approaches, Henry's fears of death overwhelm him, and he runs from the field.

FIGURE 3.1. Sample summaries

For instance, Figure 3.1 shows three sample snippets retrieved from an objective summary, interpretative summary, and synopsys for "The Red Badge of Courage," as made available by CliffsNotes.

There are a total of 66 novels, each having 108,800 words on average, where the shortest book has 16,620 words whereas the longest one has 467779 words. Furthermore, each book has 32 chapters on average. Note that the chapters are created by the authors at the time of the writing of the book. In addition, there are a total of 1109 abstracts in the entire dataset, where 442 of these abstracts are objective, 290 are interpretative summaries, and 377 are synopsis. Table 3.1 lists some of the properties of this dataset by summary publisher. It should be noted however that the majority of interest in this thesis will be given to objective summaries. Synopsis are never used in any study given in this thesis, and the interpretative summaries are only used in Chapter 4.

Publisher	N	μ_W	min_W	max_W	μ_C
Objective Summaries					
CliffsNotes	56	6807	1527	24535	94%
SparkNotes	66	7179	1600	28697	93%
MonkeyNotes	61	8783	2153	29475	91%
GradeSaver	65	8561	2406	25557	92%
NovelGuide	53	6987	2058	23880	93%
JiffyNotes	28	9669	4711	18441	87%
Barrons	36	13108	5053	20254	88%
BookRags	52	11736	2559	46480	88%
BookWolf	25	4837	2348	13292	95%
Interpretative Summaries					
CliffsNotes	53	8583	1958	23466	92%
SparkNotes	65	6241	2671	29907	94%
MonkeyNotes	61	6454	1713	12459	93%
GradeSaver	64	7805	1971	18630	92%
NovelGuide	25	4091	1192	12956	96%
BookWolf	22	4037	940	6636	96%
Synopsis					
CliffsNotes	54	1022	348	2755	99%
SparkNotes	64	967	325	2686	99%
MonkeyNotes	61	951	135	4290	99%
GradeSaver	63	1454	417	3709	99%
JiffyNotes	25	922	252	1602	99%
Barrons	36	971	422	1624	99%
BookRags	52	764	319	1916	99%
BookWolf	21	748	320	1802	99%

TABLE 3.1. Statistical properties of the different types of summaries in the dataset by publisher. N represents the number of summaries from the corresponding publisher, μ_W, min_W, and max_W represent the average, minimum, and maximum number of words in the summaries respectively, and μ_C indicates the average compression ratio.

3.2. Evaluation: ROUGE

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation [50]. It is a package that includes measures to automatically determine the quality of a given summary by comparing it to model summaries written by humans. Similar to BLEU [79], used in machine translation evaluations, the metrics in ROUGE are found to be highly correlated with human evaluations[2].

The ROUGE package includes various methods based on different measures to automatically evaluate summaries. Of these methods, ROUGE-1, ROUGE-2, and ROUGE-SU4 are the three metrics used in Document Understanding Conferences (DUC) to evaluate single document summarization systems, as these metrics are found to correlate well with human

[2]ROUGE is available at http://berouge.com

judgements. Hence the same convention is followed in this thesis. Therefore, the description of other ROUGE metrics such as ROUGE-L, ROUGE-W are skipped. The reader can refer to [50] for further details about them.

ROUGE-1, and ROUGE-2 belong to the same class of ROUGE metrics, namely ROUGE-N, which is based on n-gram cooccurence statistics, and ROUGE-SU4 is an extension of ROUGE-S. ROUGE-N measure is based on the n-gram recall between the model summaries and the candidate summary. Let R represent the set of n-grams found in a model summary, M represent the set of model summaries, and C represent the set of n-grams found in the candidate summary, then ROUGE-N metric can be formally defined as:

$$Recall^{N}_{M,C} = \frac{\sum_{R \in M} |R \cap C|}{\sum_{R \in M} |R|} \tag{1}$$

Since the denominator of this equation contains the sum of n-grams found in reference summaries, ROUGE-N is a recall-based metric. Furthermore, replacing N with 1, and 2, it should be clear that ROUGE-1 metric looks for the unigram cooccurences, and the ROUGE-2 metric looks for the bigram cooccurences in the candidate and model summaries.

One disadvantage of the ROUGE-2 measure is that it considers substrings rather than subsequences. As a result, it cannot find a match between the strings "his old book" and "his book". In order to account for this, ROUGE-S metric is introduced which computes the skip-bigram cooccurence statistics. The formula for ROUGE-S is the same as ROUGE-2 except that the number of bigram subsequences in a summary of m words is no longer $m-1$, but rather given as $C(m, 2)$ where $C(x, y)$ is the combination function.

The ROUGE-S metric still has a drawback, that is it does not distinguish between the cases where the reference and the candidate summaries have no word overlap vs. the summaries do have word some overlapping words but no overlapping skip-bigrams. In order to account for this, ROUGE-SU measure is introduced, which also calculates unigram overlaps in addition to skip-bigram overlaps. In other words, ROUGE-SU is a weighted average between ROUGE-S, and ROUGE-1.

Finally, a limit is given to the number of words in between the words of a skip-bigram. In the ROUGE-SU4 metric, this limit is 4, i.e. it is allowed to skip a maximum of 4 words to form the skip-bigrams. Hence in this measure, the beginning and the ending words of a sentence of 7 words would not form a skip-bigram as a subsequence of the sentence.

CHAPTER 4

ALIGNMENT

4.1. Introduction

This chapter addresses the first step in automatic summarization, and analyze the extent to which human-written summaries of literary novels can be obtained through extractive methods. A decomposition algorithm is used to automatically identify matches between sentences in the summary and sentences in the novel, and thus determine the potential coverage of extractive summarization.

The work is inspired by the decomposition algorithm previously proposed by [38] for single-document summarization on the newswire domain. In this chapter, the applicability of the algorithm to novels and their summaries is considered, and the effect of the various parameters of the algorithm is analyzed with respect to the coverage of the decomposition. The results suggest that different parameter settings are needed for long documents, and even under restrictive conditions a significant number of summary sentences can be obtained through cut-and-paste operations from the original novel. In turn, this coverage depends on the type of summaries being analyzed, with significant differences observed between objective (plot) summaries and interpretative summaries.

4.2. Dataset

Two kinds of summaries are targeted for the analysis in this study; the objective, and the interpretative summaries. From the entire novel data set, the novels that have both objective and interpretative summaries available from at least two publishers were selected. Moreover, an additional constraint is also placed on the length of these summaries, such that it is required that the interpretative summaries be at least as long as the corresponding objective summaries, when considering the same source. The reason for this constraint is, as the next sections will explain, the decomposition algorithm works with only surface level

syntactic features of the text, hence is directly affected by the compression ratio. Thus the more compressed a summary is, it is less likely that source sentences are directly used in it, but more likely that the source sentences are rephrased to bring them into a more compact form. Therefore the dataset gives bias to interpretative summaries rather than objective summaries. As the results will show, even with the bias, the interpretative summaries are not suitable for extraction.

The final dataset consisted of 31 books, which are used in the analysis described in this chapter. The novels in this collection have an average length of 87,000 words. The average length for the objective and interpretative summaries is 6,800 words per summary; where it is 5,840, and 8,230 words for objective and interpretative summaries respectively.

4.3. Decomposition Algorithm

In order to analyze the sentences in human-written summaries, the summary sentence decomposition methodology proposed in [38] is used. This methodology is based on the assumption that the human summarizers often extract phrases from the original source, and then make further editions to compose the summary sentence. The technique, also referred to as cut-and-paste, is supported by the studies in [23] and [27].

The goal of the decomposition analysis of a summary sentence is to discover the source sentences from which the cut-and-paste is done. Note that this could be a difficult task, as the extracted phrases of the source sentence can go through several transformations as a result of the editions that human-summarizers perform. The transformations may make the summary sentence become quite different as compared to the the phrases extracted from the source text.

4.3.1. Cut-and-Paste Operations

Six major operations performed during cut-and-paste based summarization are identified in [38] as a result of analyzing 120 sentences from 15 human-written summaries from the newswire domain. These operations are sentence reduction, sentence combination, syntactic transformation, lexical paraphrasing, generalization/specification, and reordering. It

is found that the human-summarizers often use a combination of these operations in order to come up with the summary of the information source.

Determining whether a phrase in a summary sentence is a result of lexical paraphrasing or a generalization/specification of phrase(s) of the source text is a difficult problem, and it is omitted in [38] as well as this study. Hence, in the decomposition algorithm, the only transformations considered are the sentence reduction, sentence combination, syntactic transformation, and the reordering operations.

The sentence reduction operation refers to extracting a sentence from the original source and then removing certain words or phrases from it. Sentence combination is the process of combining two or more sentences from the original source and merging them into one sentence. Note that it is possible to combine only parts of the sentences, hence this operation is often used together with the sentence reduction operation. Syntactic transformation refers to modification of the syntactic structure of a sentence, such as word reordering or passive transformations. Finally, the reordering operation is concerned with changing the position of the sentence in the summary with respect to the sentences in the source text that are used to construct it.

4.3.2. Problem Formulation

Based on the assumptions discussed in the previous section, the sentence decomposition problem translates into finding the words of a summary sentence inside the source text. If the words come from a single sentence, then it can be concluded that either the source sentence is included as-is in the summary, or that the sentence reduction operation is used to get rid of some of the words. If the position of the words is changed with respect to the source sentence, then syntactic transformations are also involved. Finally, if some of the words come from different sentences, then it can be concluded that the sentence combination operation is used. Note that if some words in a summary sentence are in a rephrased form of the words in the source document, then the algorithm will not catch this as the lexical paraphrasing, and generalization/specification operations are omitted.

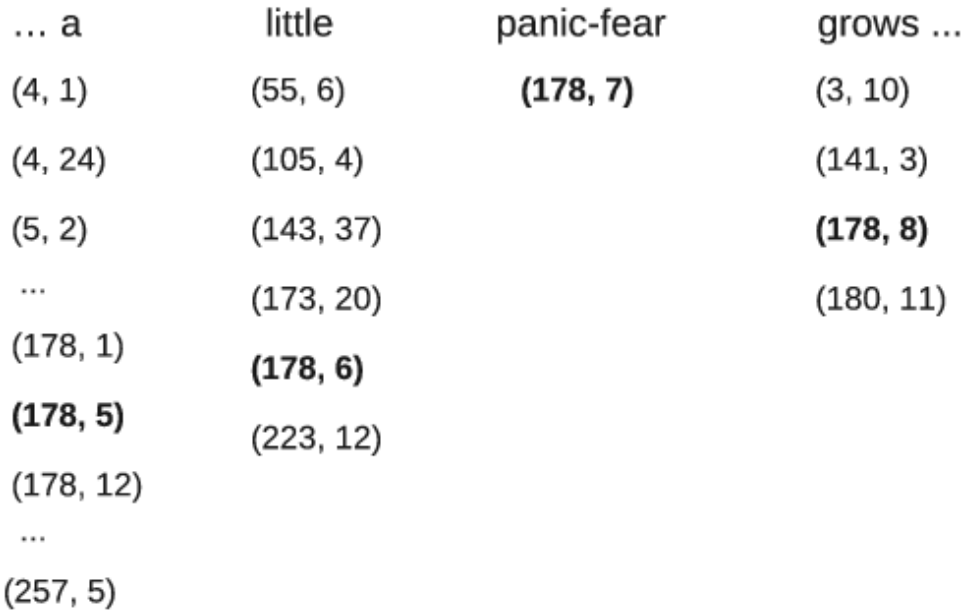

FIGURE 4.1. The document positions listed for each word of a part of a summary sentence.

Thus the problem is formulated as follows. Each summary sentence is represented as a sequence of words $(w_1, w_2, ..., w_n)$, and each word w_i can be represented as a set of tuples $S_i = (K, L)$, where K is the position of the sentence in the source document which contains w_i, and L is the position of w_i within the sentence. The tuple (K, L) is also referred to as the document position of a word. For example, the tuple $(5, 12)$ for a word w means that w appears as the 12th word of the 5th sentence of the source document. Figure 4.1 shows an example part of a sentence and the document positions listed for each word.

Hence, for a summary sentence of n words, there are $M = \prod_{i=1}^{n} |S_i|$ possible ways to compose it, where $|S_i|$ denotes the cardinality of the set S_i. The objective here is to find the set that will select the most likely document position for each word. The next section describes the algorithm that attempts to do this task in an efficient way.

4.3.3. Algorithm

The likelihood of the document position for a word can be estimated by looking at the document positions of the preceding words. This estimation can be modeled using an

n-gram model. Using a bigram model, the probability $P(w_i = (K_q, L_q)|w_{i-1} = (K_r, L_r))$ where $(K_q, L_q) \in S_i$ for $q = 1, 2, ..., |S_i|$ and $(K_r, L_r) \in S_{i-1}$ for $r = 1, 2, ..., |S_{i-1}|$, is specified as the probability of the word w_i coming from document position (K_q, L_q) given that the word w_{i-1} comes from position (K_r, R_r).

Given two adjacent words in a summary sentence, the algorithm seeks to find these two words in the source text. During this step, if the words are found in the source, then it can be concluded that the algorithm encountered one of the following steps: Both words might be found in the same sentence, placed next to each other, and following the same order. This would be the ideal case as it would imply that the words are just cut from the source sentence and pasted into the summary sentence. This case is referred to as case 1. Case 2 occurs when the words are in the same sentence, in the same order, but not necessarily consecutively as there might be other words in between. The third option, case 3, is where the words are found in the same sentence but in different order. In the fourth option, case 4, the order of the words is retained, however they come from different but neighboring sentences, where the neighboring is determined via a window parameter. Case 5 is the same as case 4 but this time the order of the words is also reversed. The final option, case 6, is the worst possible scenario and occurs when the words are found in non-neighboring sentences in any order. Figure 4.2 shows an example of each possible case using the example given in Figure 4.1.

Each document position can be seen as a state, and each of the cases defined above can be seen as a transition from one of the states of the word w_i to one of the states of the adjacent word w_{i+1} (If the adjacent word has no states, then the states of the next word, w_{i+2} are considered). These transitions can be represented using the bigram model described above, and the probabilities for the possible transitions can be defined by assigning a probability to each case. Building these states and the transition probabilities for the entire summary sentence is equivalent to constructing a first-order Hidden Markov Model (HMM). Thus it is possible to use the Viterbi algorithm to find the most likely sequence of states in this HMM. The algorithm associates a probability for the current state based on the

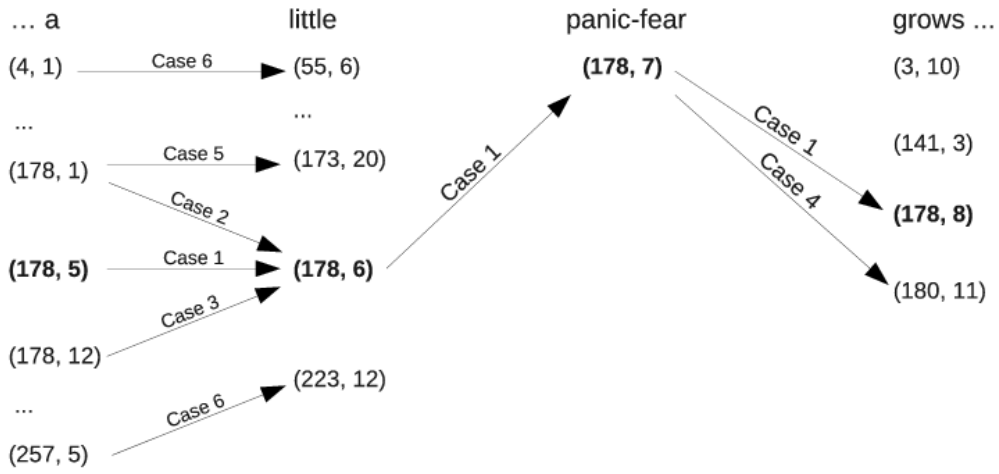

FIGURE 4.2. Illustration of cases. Note that not all the possible cases are drawn.

probability of the previous states and the transition probability from the previous state to the current state. Hence a probability for state $w_i = (K_q, L_q)$ is defined as follows:

(2)
$$P(w_i = (K_q, L_q)) = \max_{(K_r, L_r) \in S_{i-1}} P(w_{i-1} = (K_r, L_r)) \times P(w_i = (K_q, L_q) | w_{i-1} = (K_r, L_r))$$

The probability for the base case is defined as $P(w_1 = (K_q, L_q)) = 1$ for all q, to make sure that every first word of a sentence has an equal chance of being selected. Hence the probability $\max P(w_1, w_2, ..., w_n)$, which denotes the probability of the most likely sequence of document positions that make the summary sentence, can now be approximated as:

(3)
$$\max P(w_1, w_2, ..., w_n) = \max_{(K_q, L_q) \in S_n} P(w_n = (K_q, L_q))$$

In order to find the most likely sequence, as in any dynamic programming algorithm, a back pointer is kept in each step to the previous state that gives the highest probability.

For cases 1 through 5, the same transition probabilities given in [38] are used. Namely the probabilities are 1.0, 0.9, 0.8, 0.7, and 0.6 for case 1, case 2, case 3, case 4, and case 5 respectively. The probability of case 6 is used as a parameter, and varied during the analysis.

Note that these values are not tuned for a particular task, but rather just assigned intuitively in an ad-hoc fashion.

4.4. Parameter Analysis for the Decomposition Algorithm

The dataset used in [38] is composed of news articles and their summaries, and most of the parameters were selected intuitively based on this data genre. In this study, rather than taking the same parameters for granted, the goal is to see the effects of those parameters when the decomposition algorithm is applied on novel summaries. Therefore having described the algorithm, the first step is to describe the parameters, and then the analysis is performed next. Note that the intention here is not to tune these parameters, but rather to see how the decomposition algorithm is affected by different parameter choices.

Three parameters, whose values can have an effect on the algorithm, will be analyzed. First, the window size parameter w, which is the number of neighboring sentences that can be considered during a sentence combination operation alongside the current source sentence. Second, the probability p which is assigned to the transition which occurs when a word is found outside the current source sentence, and outside the neighboring sentences that are within the window size. Since the novels are long documents and their summaries have lower compression ratios[1], it can be expected that human summarizers combine sentences that are further apart from each other compared to news articles when forming the summary. Hence it should be expected that both w and p will differ for novels. Furthermore, note that p and w are dependent on each other such that decreasing (increasing) the window size will increase (decrease) the chances of finding the next word outside of the neighboring sentences.

Finally, the last parameter is a rule that specifies the number of stop words, sw, and/or the number of non-stop words, nsw, that need to be found in a document (source) sentence in order to consider that sentence as being involved in the cut-and-paste process. The values for these parameters used in [38] are $w = 3$, $p = 0.5$, and for the final rule, it was enforced that a document sentence has to contribute to the summary sentence with either

[1]Recall from Chapter 1 that a lower compression ratio means a higher compression of the source text.

two or more non-stop words or a non-stop word plus one or more stop words, which in logical form can be written as $(nsw \geq 2) \vee (nsw \geq 1 \wedge sw \geq 1)$.

Furthermore, it is assumed in [38] that a summary sentence is formed as a result of cut-and-paste operations on source sentences if and only if at least half of the words in the summary sentence can be found in the source sentences. Rather than using the same assumption, in this study, a coverage for each summary sentence is computed, which is simply the percentage of the words in the summary sentence that are found in the source document. Using this notion, it is possible to create the plots of coverage for each summary-document pair, and visualize the effect of varying coverage. In addition, this method also allows one to specify a cutoff value for the coverage and discard the sentences that fall below the cutoff. Note that when the cutoff value is equal to 0.5, the assumption would be the same as in [38].

It is also possible to parametrize the other transition probabilities from case 1 to case 5, and analyze their effects individually as these probabilities are assigned intuitively in an ad-hoc fashion. However the resulting analysis would not give us any insights for novels. Therefore in order to be able to compare the results with the newswire domain, those probabilities are left unchanged.

4.4.1. Number of Words in Contributing Document Sentences

The first parameter analyzed is the number of stop words and non-stop words in a contributing document sentence. Note that even at chapter level, the novels have significantly greater length than news articles. Since there is a substantially larger number of sentences, the probability of finding a summary word or phrase in multiple sentences is also greatly increased. Therefore in the experiments, the conditions are gradually made more restrictive. During the analysis of this parameter, the other parameters w, and p are kept fixed to 3 and 0.5 respectively, which are the values used previously for news articles.

As the least restrictive rule, the experiments start with a simple condition, $(nsw \geq 1)$. The next rules in order of strength are; $(nsw \geq 2) \vee (nsw \geq 1 \wedge sw \geq 1)$, $(nsw \geq 2)$, $(nsw \geq 3) \vee (nsw \geq 2 \wedge sw \geq 1)$, $(nsw \geq 3)$, $(nsw \geq 4) \vee (nsw \geq 3 \wedge sw \geq 1)$, and $(nsw \geq 4)$.

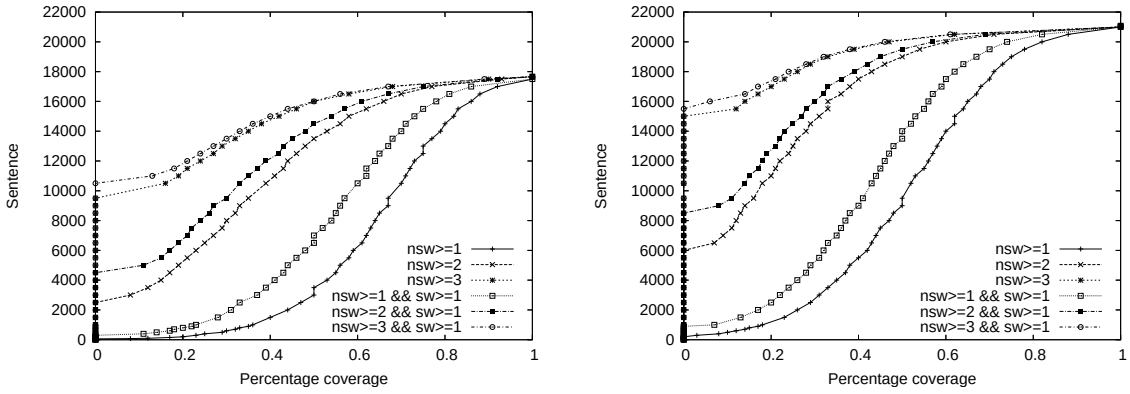

FIGURE 4.3. (a) Objective summaries (b) Interpretative summaries

The results are shown in Figure 4.3. The figure plots the coverage for both the objective and the interpretative summaries in the dataset. For each summary sentence, its coverage is measured as the percentage of words in the sentence that are obtained from the source text through the cut-and-paste operations. The Y axis shows all the sentences for all the summaries in the dataset, in increasing order of their coverage. 17,665 objective and 21,077 interpretative summary sentences from 55 sources are analyzed in the experiments.

For both the objective and the interpretative summaries, increasing the restrictiveness of the conditions results in less coverage. Furthermore, although the results are similar for both objective and interpretative summaries, as the coverage rate increases, the drop in the number of sentences is much quicker for the interpretative summaries. In order to see this difference, a set of varying cutoff values are applied on these results and the percentage of the sentences that exceed the cutoff value is reported. This value represents the percentage of the sentences in the summary that are constructed by cut-and-paste operations for the given parameter choices.

The difference between the objective and the interpretative summaries is quite clear from the results in Table 4.1. For example, by applying the second rule and a cutoff of 0.5, which are the same assumptions as in [38], 65.5% of the objective summary sentences can be identified as constructed by cut-and-paste operations. Using the same assumptions, the number dramatically reduces to 37.8% for the interpretative summaries. For the remainder

Rules	Cutoff					
	0.2	0.3	0.4	0.5	0.6	0.7
Objective summaries						
$(nsw \geq 1)$	98.9	96.7	92.4	83.9	66.0	41.9
$(nsw \geq 2) \vee (nsw \geq 1 \wedge sw \geq 1)$	95.9	90.0	80.0	65.5	42.1	20.8
$(nsw \geq 2)$	71.7	55.6	39.7	26.3	14.2	0.69
$(nsw \geq 3) \vee (nsw \geq 2 \wedge sw \geq 1)$	62.2	46.3	31.7	20.1	10.6	0.54
$(nsw \geq 3)$	37.2	25.9	17.0	10.9	0.61	0.35
$(nsw \geq 4) \vee (nsw \geq 3 \wedge sw \geq 1)$	33.6	23.6	15.6	10.0	0.57	0.33
$(nsw \geq 4)$	19.9	15.0	10.4	0.71	0.42	0.26
Interpretative summaries						
$(nsw \geq 1)$	94.7	87.5	74.9	57.7	35.0	17.0
$(nsw \geq 2) \vee (nsw \geq 1 \wedge sw \geq 1)$	88.5	75.5	57.5	37.8	18.2	0.78
$(nsw \geq 2)$	48.2	30.7	17.9	10.2	0.53	0.30
$(nsw \geq 3) \vee (nsw \geq 2 \wedge sw \geq 1)$	40.5	24.7	14.1	0.81	0.44	0.27
$(nsw \geq 3)$	19.5	12.1	0.73	0.48	0.31	0.20
$(nsw \geq 4) \vee (nsw \geq 3 \wedge sw \geq 1)$	17.8	11.2	0.68	0.46	0.30	0.20
$(nsw \geq 4)$	10.5	0.75	0.51	0.37	0.26	0.18

TABLE 4.1. Percentage of sentences in the objective and interpretative summaries that are constructed by cut-and-paste operations, as a function of cutoff value and word restriction rules.

of the analysis reported in this section, for consistency purposes, this parameter is fixed to the second rule, $(nsw \geq 2) \vee (nsw \geq 1 \wedge sw \geq 1)$.

4.4.2. Probability Assigned to Distant Words

The parameter p is analyzed next, which specifies the bias of the algorithm to consider combinations of phrases from sentences that are distant from each other, where the distance is defined by the window-size w. For instance, when p is set to 0.1, the algorithm will assign a probability value of 0.1 to the transitions occurring as a result of case 6. In order to see the effect of this bias, p is varied in 0.1 increments from 0.0 to 0.5 while keeping the other parameters fixed.

The results are plotted in Figure 4.4. Once again, the coverage of the summary sentences are shown for both the objective and the interpretative summaries in the dataset. A summary of the results are also displayed in Table 4.2, which shows the percentage of

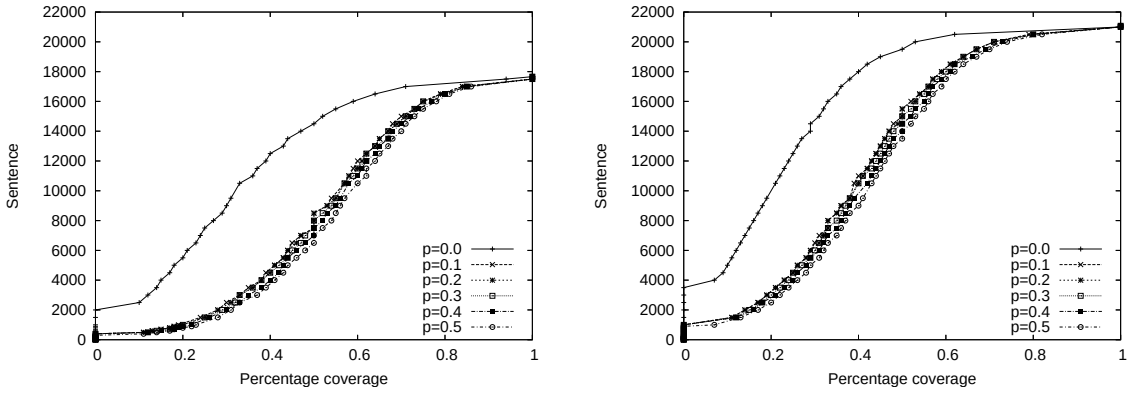

FIGURE 4.4. (a) Objective summaries (b) Interpretative summaries

sentences in the objective and interpretative summaries that are constructed by cut-and-paste operations.

As seen in the figure and the table, the coverage is greatly affected when p is reduced to 0, which only allows for the combination of sentences found within the specified window size. This suggests that there is a substantial amount of sentences in the summary that could be formed by the combination of distant sentences. However, it should be noted that most of these sentences are most likely not constructed using the cut-and-paste techniques but rather the words in the summary sentences are just found randomly in the source text as even a small increment in p, results in a very big jump on the results. Nevertheless, it can also be seen from Table 4.2 that for the objective summaries, when p is set to 0, and the cutoff is set to 0.5, 18.8% of the summary sentences are formed by using the cut-and-paste operations. In this case, the randomness factor is extremely reduced, as the parameters are quite restrictive.

4.4.3. Window Size

The final parameter that will be investigated is the window size, w. As mentioned, this parameter is related to p, so with increasing w it is expected that some of the sentences that are lost with the reduction of p will be regained. Thus, for this experiment the value of p will be fixed to 0, and w will be gradually increased by setting it successively to $3, 9, 15, 21, 27$. The results are presented in Figure 4.5.

Probability	Cutoff					
	0.2	0.3	0.4	0.5	0.6	0.7
Objective summaries						
$p = 0.0$	70.1	49.4	31.1	18.8	0.91	0.44
$p = 0.1$	94.3	86.5	74.2	57.8	34.6	16.3
$p = 0.2$	94.4	86.8	74.6	58.4	35.1	16.6
$p = 0.3$	94.8	87.5	75.9	56.9	36.5	17.2
$p = 0.4$	95.4	88.7	77.7	62.5	38.8	18.8
$p = 0.5$	95.9	90.0	80.0	65.5	42.1	20.8
Interpretative summaries						
$p = 0.0$	54.0	30.8	15.6	0.75	0.34	0.20
$p = 0.1$	85.5	70.3	50.0	30.7	14.1	0.60
$p = 0.2$	85.7	70.7	50.6	31.2	14.4	0.61
$p = 0.3$	86.3	71.8	52.2	32.5	15.1	0.64
$p = 0.4$	87.4	73.4	54.8	34.9	16.5	0.69
$p = 0.5$	88.5	75.5	57.5	37.8	18.2	0.78

TABLE 4.2. Percentage of sentences in the objective and interpretative summaries that are constructed by cut-and-paste operations varying with cutoff and p.

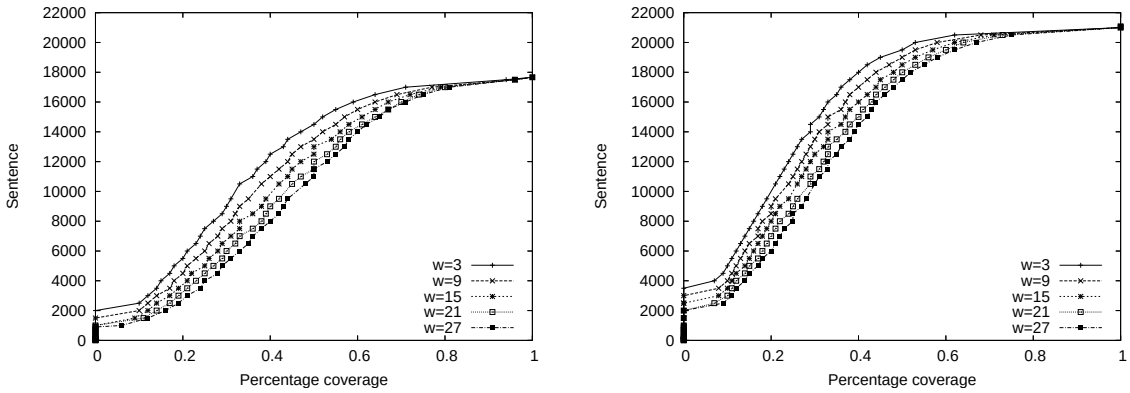

FIGURE 4.5. (a) Objective summaries (b) Interpretative summaries

As before, the results with the specific cutoff values applied are shown in Table 4.3. Increasing the window size allows the algorithm to combine the sentences that are distant from each other, which is an expected result. Almost a linear increment is observed on the coverage with the increasing window size. Note that, even with an unrealistically large window size of 27, the coverage reaches to only 39% for objective summaries with a cutoff value of 0.5, which is quite below 57.8% achieved with p set to 0.1 with the same cutoff value.

Probability	Cutoff					
	0.2	0.3	0.4	0.5	0.6	0.7
Objective summaries						
$w = 3$	70.1	49.4	31.1	18.8	0.91	0.44
$w = 9$	75.3	56.6	39.0	25.2	13.2	0.65
$w = 15$	79.5	62.6	45.7	30.8	16.8	0.82
$w = 21$	82.5	67.3	51.1	35.6	19.9	0.96
$w = 27$	85.4	71.5	55.3	39.8	22.5	10.7
Interpretative summaries						
$w = 3$	54.0	30.8	15.6	0.75	0.34	0.20
$w = 9$	59.8	37.0	20.5	10.8	0.48	0.26
$w = 15$	64.3	42.3	25.1	13.8	0.62	0.30
$w = 21$	68.3	47.2	29.3	16.4	0.76	0.35
$w = 27$	71.6	51.3	32.8	18.8	0.87	0.38

TABLE 4.3. Percentage of sentences in the objective and interpretative summaries that are constructed by cut-and-paste operations varying with cutoff and p.

This shows that, most of the increase in coverage when p is non-zero, comes from finding the words of the summary sentence in random sentences in the source text.

4.4.4. The Decomposition of Novel Summaries

In an analysis of 1,642 summary sentences in [38], it is found out that 42% of these sentences match to a single sentence in the document, 36% of them match to two sentences, and only 3% match to three or more sentences. It was concluded that 78% of the summary sentences are constructed by cut-and-paste operations by only counting the ones that use 1 or 2 document sentences. The study was however limited to news articles. In comparison, for novel summaries, even at chapter level, the compression rates are quite smaller, so that it would be quite reasonable for a novel summary sentence to use three or more document sentences. See for instance the following example, which is a sentence from a human summary for Moby Dick by Herman Melville, along with three sentences in the source text that contribute to this summary sentence.

SUMMARY SENTENCE (51 WORDS): Flask is the last person down at the table and the first one to leave; since Flask had become an officer he had never known what it was to be otherwise than hungry, more or less, for what he eats does not relieve his hunger as keep it immortal in him.

BOOK SENTENCES:
Sentence 33 (8 words): Flask was **the last person down at the** dinner, and Flask is **the first** man up.
Sentence 38 (20 words): Therefore it was that Flask once admitted in private, that ever **since** he **had** arisen to the dignity of **an officer**, from that moment **he had never known what it was to be otherwise than hungry, more or less**.
Sentence 39 (13 words): **For what he** ate did **not** so much **relieve his hunger, as keep it immortal in him**.

FIGURE 4.6. Sample summary sentence and the three source sentences used to compose it

Not only the above example demonstrates the fact that a summary sentence can easily be constructed from three document sentences, but it also shows that a larger window-size than 3 is quite possible for long documents.

As mentioned, 17,665 objective, and 21,077 interpretative summary sentences from 55 sources are analyzed in the experiments. For the purpose of comparison of the results with the ones in [38], the parameters are fixed to $w = 3$, $p = 0.5$, and the rule $(nsw \geq 2) \vee (nsw \geq 1 \wedge sw \geq 1)$. The results are shown in Table 4.4. For objective summaries, 34.5% of the sentences cannot make the cutoff and hence are not identified as constructed by cut-and-paste operations. On the other hand, 48.4% of the sentences use 1, 2, 3, or 4 document sentences, and 17.1% of them use 5 or more. For interpretative summaries, 62.2% of them cannot make the cutoff, and 25.3% use 1, 2, 3, or 4 document sentences, and 12.5% use 5 or more.

Although the objective summaries are clearly more suitable for extractive summarization than the interpretative ones, it is quite unlikely that 48.4% of the objective summary sentences can be obtained through the cut-and-paste process using up to 4 document sentences as this result is obtained by fixing the parameters to those tuned for news articles.

	Number of document sentences used				
Summary type	1	2	3	4	5 or more
Objective	5.3%	13.8%	15.6%	13.7%	17.1%
Interpretative	3.1%	6.3%	8.6%	7.3%	12.5%

TABLE 4.4. Percentage of the number of document sentences used to compose the summary sentences

Therefore it should be expected that there is some noise in the results due to the looser parameters. As it was seen in the experiments, increasing the window size w to even unrealistic values still does not make up for the reduction obtained when the parameter p is set to 0. Therefore, keeping p at 0, and increasing the window size to 9, which is more realistic, it can be seen from Table 4.3 that, a coverage cutoff value of 0.4 gives 39%, and a value of 0.5 gives 25.2%. Given that these parameters are far more reasonable for novels, it is reasonable to expect the actual number to be in this range.

4.5. Related Work

In terms of analysis of the composition of human-written summaries, the work most closely related to the one described in this chapter is the decomposition algorithm proposed in [39, 38], which analyzed the extent to which single-document summaries of news articles are created by using cut-and-paste operations from the text. More recently, their technique has been applied to the analysis of human summaries of Japanese broadcast news [96], and has also been adapted for the analysis of multi-document human summaries [2].

Another document/abstract alignment approach is given in [18, 19], in which a more sophisticated approach than the standard HMM model is used. The new model, phrase-based HMM, which improves upon [38], is also unsupervised, and uses Dirichlet Priors rather than the maximum likelihood estimates. It also recognizes paraphrasings in the summary to a degree. However this method is not implemented for these experiments as it lacks the speed to run it efficiently on novel summaries.

4.6. Summary

In this chapter, the sentence decomposition algorithm proposed by [38] for the newswire domain is applied to novel summaries, and the extent to which human-written novel summaries can be obtained through cut-and-paste operations from the original source is analyzed. Specifically, two separate chapter-level summary sources for novels are considered: one that attempts to give a summary by redescribing the events in a compact form, which is referred to as objective summaries, and one that attempts to give a summary by capturing the deep meaning of the story by describing the author's ideas and thoughts, which is referred to as interpretative summaries.

The result of the analysis consistently suggests that, the percentage of the objective summary sentences found by the algorithm to be constructed using cut-and-paste techniques is about twice the size of the percentage found for the interpretative summaries. Even with the varying parameters the argument still holds valid, except for cases where the parameters are unrealistically restrictive. It can therefore be concluded that humans use very little extraction from the source document when writing interpretative summaries, and thus extractive summarization is non-suitable for this summary type. On the other hand, with reasonable parameter choices, around 30% of the human-written objective summary sentences are constructed from the original document on average, which indicates that extractive summarization is more suitable for this summary type.

The results differ from those reported in [38], where the analysis is performed on short articles from the news domain. Using the same parameters, the decomposition algorithm fails to find a match for 34.5% of the objective summary sentences compared to only 19% in [38]. Although some of those sentences are still constructed by cut-and-paste, the algorithm fails to find them due to other transformations applied to these sentences. Part of this is most likely caused due to the fact that heavy editing operations such as paraphrasing or generalization/specification are more frequently encountered in the construction of sentences in book summaries due to the low compression ratios. Another effect of the lower compression ratio is seen in the number of source sentences used to construct the novel summaries.

The average compression ratio per chapter was 7.5% for objective summaries, and 11% for interpretative summaries, compared to a ratio around 20-30% that is typical for the newswire domain.

A decomposition analysis for literary novel summaries not only helps us make the distinction between two summary styles (e.g., objective vs. interpretative), but it also helps us see how humans transform the document sentences into summary sentences, and which document sentences are selected for inclusion into a summary through cut-and-paste operations.

CHAPTER 5

ANALYSIS OF EXTRACTIVE SUMMARIZATION

5.1. Introduction

This chapter analyzes the topic identification stage of single-document automatic text summarization across four different domains (genres), consisting of newswire, literary, scientific and legal documents. It presents a study that explores the summary space of each domain via an exhaustive search strategy, and finds the probability density function (pdf) of the ROUGE score distributions. The resulting pdfs are then used to calculate the percentile rank of extractive summarization systems. The results introduce a new way to judge the success of automatic summarization systems and bring quantified explanations to questions such as why it was so hard for the systems to date to have a statistically significant improvement over the lead baseline in the news domain.

Since the sentences in a document are reproduced verbatim in extractive summaries, it is theoretically possible to explore the search space of this problem through an enumeration of all possible extracts for a document. Such an exploration would not only allow us to see how far one can go with extractive summarization, but it would also make it possible to judge the difficulty of the problem by looking at the distribution of the evaluation scores for the generated extracts. Moreover, the high scoring extracts could also be used to train a machine learning algorithm.

However, such an enumeration strategy has an exponential complexity as it requires all possible sentence combinations of a document to be generated, constrained by a given word or sentence length. Thus the problem quickly becomes impractical as the number of sentences in a document increases and the compression ratio decreases. In this work, an attempt is made to overcome this bottleneck by using a large cluster of computers, and decomposing the task into smaller problems by using the given section boundaries or a linear

text segmentation method. As a result of this exploration, a probability density function (pdf) of the ROUGE score [50] distributions is generated for four different domains, which shows the distribution of the evaluation scores for the generated extracts, and allows one to assess the difficulty of each domain for extractive summarization.

Furthermore, using these pdfs, a new success measure is introduced for extractive summarization systems. Namely, given a system's average score over a data set, calculation of the percentile rank of this system from the corresponding pdf of the data set is shown. This allows one to see the true improvement a system achieves over another, such as a baseline, and provides a standardized scoring scheme for systems performing on the same data set.

5.2. Related Work

Despite the large amount of work in automatic text summarization, there are only a few studies in the literature that employ an exhaustive search strategy to create extracts, which is mainly due to the prohibitively large search space of the problem. Furthermore, the research regarding the alignment of abstracts to original documents has shown great variations across domains [44, 97, 61, 38, 10], which indicates that the extractive summarization techniques are not applicable to all domains at the same level.

In order to automate the process of corpus construction for automatic summarization systems, [61] used exhaustive search to generate the best extract from a given (abstract, text) tuple, where the best extract contains a set of clauses from text that have the highest similarity to the given abstract.

In addition, [21] used exhaustive search to create all the sentence extracts of length three starting with 15 TREC Documents, in order to judge the performance of several summary evaluation measures suggested in their paper.

Finally, the study most similar to the one given in this chapter is presented in [52], which uses the articles with less than 30 sentences from the Document Understanding Conference (DUC) 2001 data set to find oracle extracts of 100 and 150 (± 5) words. These extracts were compared against one summary source, selected as the one that gave the highest inter-human agreement. Although it was concluded that a 10% improvement was possible

for extractive summarization systems, which typically score around the lead baseline, there was no report on how difficult it would be to achieve this improvement, which was the main objective and contribution of the work in this chapter.

5.3. Dataset

The data set used in this chapter is composed of four different domains: newswire, literary, scientific and legal. 50 documents and only one summary for each document are used for all domains, except for newswire where two summaries per document are used. For the newswire domain, the articles and their summaries are selected from the DUC 2002 data set[1]. For the literary domain, 10 novels are randomly drawn from the pool described in Chapter 3. Furthermore, 5 chapters are selected from each novel, and each chapter is treated as a separate document. Thus 50 chapters in total are evaluated. Recall from Chapter 3 that these chapters are not defined by an automatic segmentation algorithm, but rather by the authors of the novels at the time of their writing. For the scientific domain, the articles are selected from the medical journal Autoimmunity Reviews[2], and their abstracts are used as summaries. Finally, for the legal domain, 50 law documents and their corresponding summaries are gathered from the European Legislation Website,[3] which comprises four types of laws - Council Directives, Acts, Communications, and Decisions over several topics, such as society, environment, education, economics and employment.

Although all the summaries are human generated abstracts for all the domains, it is worth mentioning that the documents and their corresponding summaries exhibit a specific writing style for each domain, in terms of the vocabulary used and the length of the sentences. Some of the statistical properties of each domain are listed in Table 5.1.

5.4. Experimental Setup

As mentioned in Section 5.1, an exhaustive search algorithm requires generating all possible sentence combinations from a document, and evaluating each one individually. For

[1]http://www-nlpir.nist.gov/projects/duc/data.html
[2]http://www.elsevier.com/wps/product/cws_home/622356
[3]http://eur-lex.europa.eu/en/legis/index.htm

Domain	μ_{Dw}	μ_{Sw}	μ_R	μ_C	μ_{Cw}
Newswire	641	101	84%	1	641
Literary	4973	1148	77%	6	196
Scientific	1989	160	92%	9	221
Legal	3469	865	75%	18	192

TABLE 5.1. Statistical properties of the data set. μ_{Dw}, and μ_{Sw} represent the average number of words for each document and summary respectively; μ_R indicates the average compression ratio; and μ_C and μ_{Cw} represent the average number of sections for each document, and the average number of words for each section respectively.

example, using the values from Table 5.1, and assuming 20 words per sentence, one can find that the search space for the news domain contains approximately $\binom{32}{5} \times 50 = 10,068,800$ summaries. The same calculation method for the scientific domain gives $\binom{99}{8} \times 50 = 8.56 \times 10^{12}$ summaries. Obviously the search space gets much bigger for the legal and literary domains due to their larger text size.

In order to be able to cope with such a huge search space, the first thing done was to modify the ROUGE 1.5.5[4] Perl script by fixing the parameters to those used in the DUC experiments,[5] and also by modifying the way it handles the input and output to make it suitable for streaming on the cluster.

The resulting script evaluates around 25-30 summaries per second on an Intel 2.33 GHz processor. Next, for each (document, summary) pair, the resulting ROUGE script is run as part of a streaming job on a large cluster of computers running on a Hadoop Map-Reduce framework.[6] Based on the size of the search space for a (document, summary) pair, the number of computers allocated in the cluster ranged from just a few to more than one thousand.

Although the combination of a large cluster and a faster ROUGE is enough to handle most of the documents in the news domain in just a few hours, a simple calculation shows that the problem is still impractical for the other domains. Hence for the scientific, legal,

[4]http://berouge.com
[5]-n 2 -x -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0
[6]http://hadoop.apache.org/

and literary domains, rather than considering each document as a whole, the documents are divided into sections, and the extracts are created separately for each section with the constraint that the length of the extract is proportional to the length of the section in the original document. For the legal and scientific domains, the given section boundaries (without considering the subsections for scientific documents) are used to chunk the the documents into sections. For the novels, each chapter is treated as a single document (since each chapter has its own summary), and is then divided into sections using a publicly available linear text segmentation algorithm by [99].[7] In all cases, the algorithm picks the number of segments automatically.

To evaluate the sections, ROUGE is modified further so that it applies the length constraint to the extracts only, not to the model summaries. This is due to the fact that the extracts of each section are evaluated individually against the whole model summary, which is larger than the extract. This way, an overall ROUGE recall score for a document extract can be obtained by simply summing up the recall scores of the extracts of each section. The precision score for the entire document can also be found by adding the weighted precision scores for each section, where the weight is proportional to the length of the section in the original document. In this study, however, only the recall scores are used.

Note that, since for the legal, scientific, and literary domains each section of a document is considered independently, the process is not the same as performing a true exhaustive search for these domains, but is rather equivalent to solving a suboptimal problem, as the number of words in the model summary are divided to each section proportional to the section's length. This is however a fair assumption, as it has been shown repeatedly in the past that text segmentation helps improving the performance of text summarization systems [104, 72, 67].

[7] http://mastarpj.nict.go.jp/ mutiyama/software/textseg/textseg-1.211.tar.gz

5.5. Exhaustive Search Algorithm

Let $E_{i_k} = S_{i_1}, S_{i_2}, ..., S_{i_k}$ be the i^{th} extract that has k sentences, and generated from a document D with n sentences $D = S_1, S_2, \ldots, S_n$. Further, let $len(S_j)$ give the number of words in sentence S_j. We enforce that E_{i_k} satisfies the following constraints:

$$\begin{aligned} len(E_{i_k}) &= len(S_{i_1}) + \ldots + len(S_{i_k}) \geq L \\ len(E_{i_{k-1}}) &= len(S_{i_1}) + \ldots + len(S_{i_{k-1}}) < L \end{aligned}$$

where L is the length constraint on all the extracts of document D. Note that for any E_{i_k}, the order of the sentences in $E_{i_{k-1}}$ does not affect the ROUGE scores, since only the last sentence may be chopped off due to the length constraint.[8] Hence, the algorithm starts generating sentence combinations $\binom{n}{r}$ in lexicographic order, for $r = 1...n$, and for each combination $E_{i_k} = S_{i_1}, S_{i_2}, ..., S_{i_k}$ where $k > 1$, it generates additional extracts E'_{i_k} by successfully swapping S_{i_j} with S_{i_k} for $j = 1, ..., k-1$ and checking to see if the above constraints are still satisfied. Therefore from a combination with k sentences that satisfies the constraints, up to $k-1$ additional extracts might be generated. Finally, the process stops either when $r = n$ and the last combination is generated, or the algorithm cannot find any extract that satisfies the constraints for r.

5.6. Generating PDFs

Once the extracts for a document are generated and evaluated, the recall score of each result is assigned to a range, which will be referred to as a bin here. $1,000$ equally spaced bins are used between 0 and 1. As an example, a recall score of 0.46873 would be assigned to the bin $[0.468, 0.469]$. Keeping a count for each bin is in fact equivalent to building a histogram of scores for the document. Let this histogram be h, and $h[j]$ be the value in the j^{th} bin of the histogram. Then the normalized histogram $\widehat{h}$ is defined as:

[8]Note that the coherence of extracts are not taken into account, i.e. the sentences in an extract do not need to be sorted in order of their appearance in the original document. The position of the words in a sentence are also not changed.

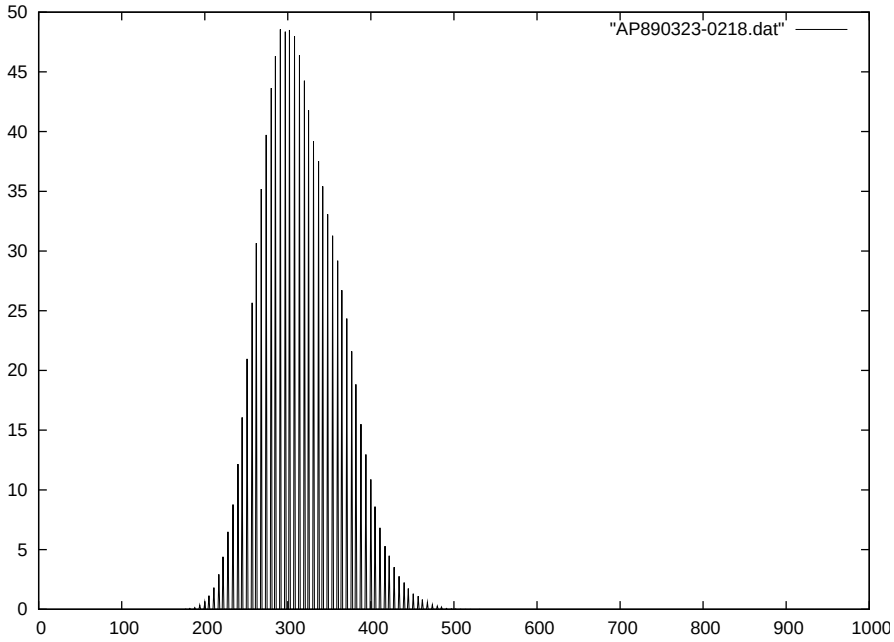

FIGURE 5.1. The normalized histogram $\widehat{h}$ of ROUGE-1 recall scores for the newswire document AP890323-0218.

$$\widehat{h}[j] = \frac{N}{\sum_{i=1}^{N} h[j]}\ h[j] \tag{4}$$

where $N = 1,000$ is the number of bins in the histogram. Note that since the width of each bin is $\frac{1}{N}$, the Riemann sum of the normalized histogram $\widehat{h}$ is equal to 1, so $\widehat{h}$ can be used as an approximation to the underlying pdf. As an example, the histogram $\widehat{h}$ for the newswire document AP890323-0218 is shown in Figure 5.1.

The normalized histograms of all the documents in a domain are combined in order to find the pdf for that domain. This requires multiplying the value of each bin in a document's histogram, with all the other possible combinations of bin values taken from each of the remaining histograms, and assigning the result to the average bin for each combination. This can be done iteratively by keeping a moving average. This procedure is illustrated in Algorithm 1, where K represents the number of documents in a domain.

The resulting histogram h_d, when normalized using Equation 4, is an approximation to the pdf for domain d. Furthermore, the *round*() function is used in line 9, which rounds a

Algorithm 1 Combine $\widehat{h}^i$'s for $i = 1, \ldots, K$ to create h_d, the histogram for domain d.

```
1: h_d := {}
2: for i = 1 to N do
3:    h_d[i] := ĥ^1[i]
4: end for
5: for i = 2 to K do
6:    h_t = {}
7:    for j = 1 to N do
8:       for k = 1 to N do
9:          a = round(((k * (i − 1)) + j)/i)
10:         h_t[a] = h_t[a] + (h_d[k] * ĥ^i[j])
11:      end for
12:   end for
13:   h_d := h_t
14: end for
```

number to the nearest integer, as the bins are indexed by integers. Note that this rounding introduces an error, which is distributed uniformly due to the nature of the *round*() function. It is also possible to lower the affect of this error with higher resolutions (i.e. larger number of bins). A sample h_d is shown in Figure 5.2, which is obtained by combining 10 documents from the newswire domain.

Recall from Section 5.4 that the documents in the literary, legal, and scientific domains are divided into sections either by using the given section boundaries or by applying a text segmentation algorithm, and the extracts of each section are then evaluated individually. Hence for these domains, the histogram of each section is calculated individually first, and then the histograms are combined to find the overall histogram of a document. The combination procedure for the section histograms is similar to Algorithm 1, except that in this case a moving average is not kept, but rather the bins of the sections are summed. Note that when bin i and j are added, the resulting values should be expected to be half the times in bin $i + j$, and half the times in $i + j - 1$.

5.7. Calculating Percentile Ranks

Given a pdf for a domain, the success of a system having a ROUGE recall score of S could be simply measured by finding the area bounded by S. This gives us the percentile

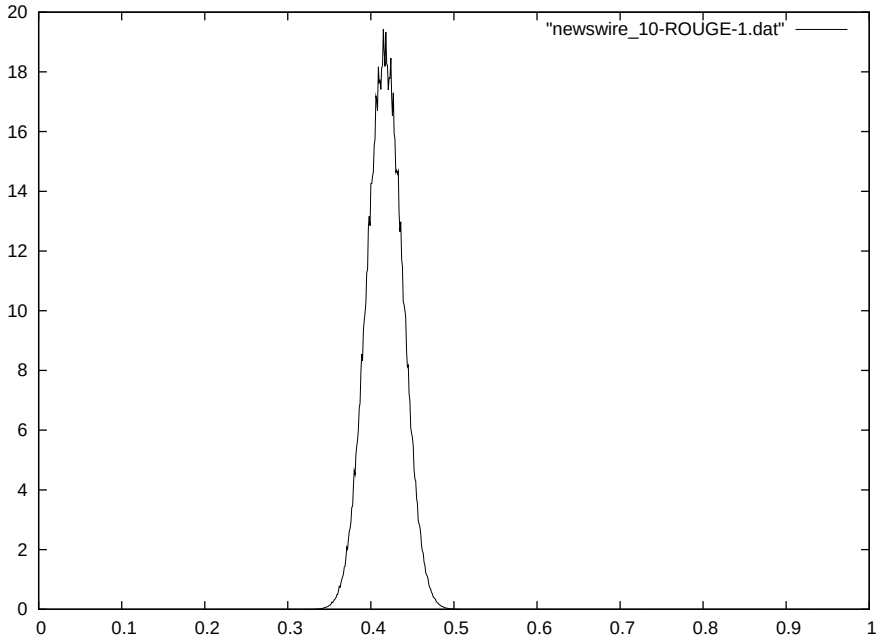

FIGURE 5.2. An example pdf obtained by combining 10 document histograms of ROUGE-1 recall scores from the newswire domain. The x-axis is normalized to [0,1].

rank of the system in the overall distribution. Assuming $0 \leq S \leq 1$, let $\widehat{S} = \lfloor N \times S \rfloor$, then the formula to calculate the percentile rank can be simply given as:

$$PR(S) = \frac{100}{N} \sum_{i=1}^{\widehat{S}} \widehat{h}_d[i] \tag{5}$$

5.8. Results and Discussion

The ensemble distributions of ROUGE-1 recall scores per document are shown in Figure 5.3. The ensemble distributions show that the performance of the extracts, especially for the news and the scientific domains, are mostly uniform for each document. This is due to the fact that documents in these domains, and their corresponding summaries, are written with a certain conventional style. There is however a little scattering in the distributions of the literary and the legal domains. This is an expected result for the literary domain, as there is no specific summarization style for these documents, but somehow surprising for the

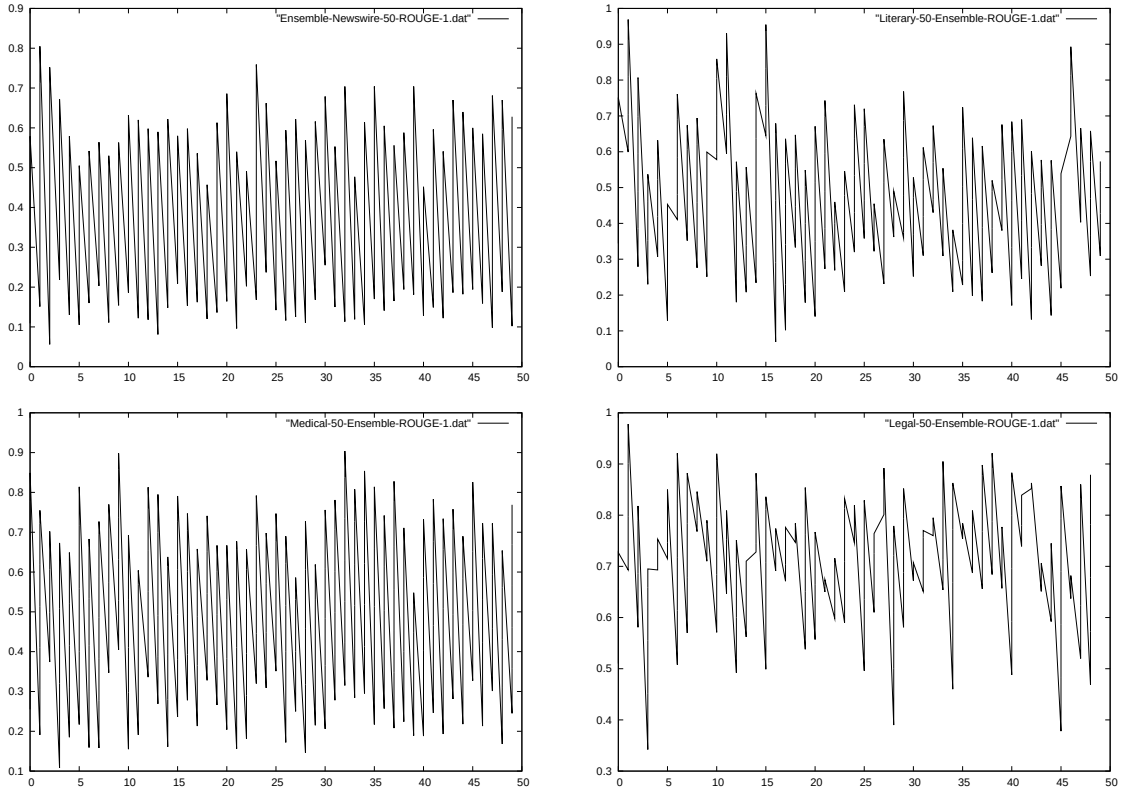

FIGURE 5.3. ROUGE-1 recall score distributions per document for Newswire, Literary (top), Scientific and Legal Domains (bottom), respectively from left to right.

legal domain, where the effect is probably due to the different types of legal documents in the data set.

The pdf plots resulting from the ROUGE-1 recall scores are shown in Figure 5.4.[9] In order to analyze the pdf plots, and better understand their differences, Table 5.2 lists the mean (μ) and the standard deviation (σ) measures of the pdfs, as well as the average minimum and maximum scores that an extractive summarization system can get for each domain.

By looking at the pdf plots and the minimum and maximum columns from Table 5.2, it can be noticed that for all the domains, the pdfs are long-tailed distributions. This immediately implies that most of the extracts in a summary space are clustered around the mean, which means that for automatic summarization systems, it is very easy to get scores

[9]Similar pdfs are obtained for ROUGE-2 and ROUGE-SU4, even if at a different scale.

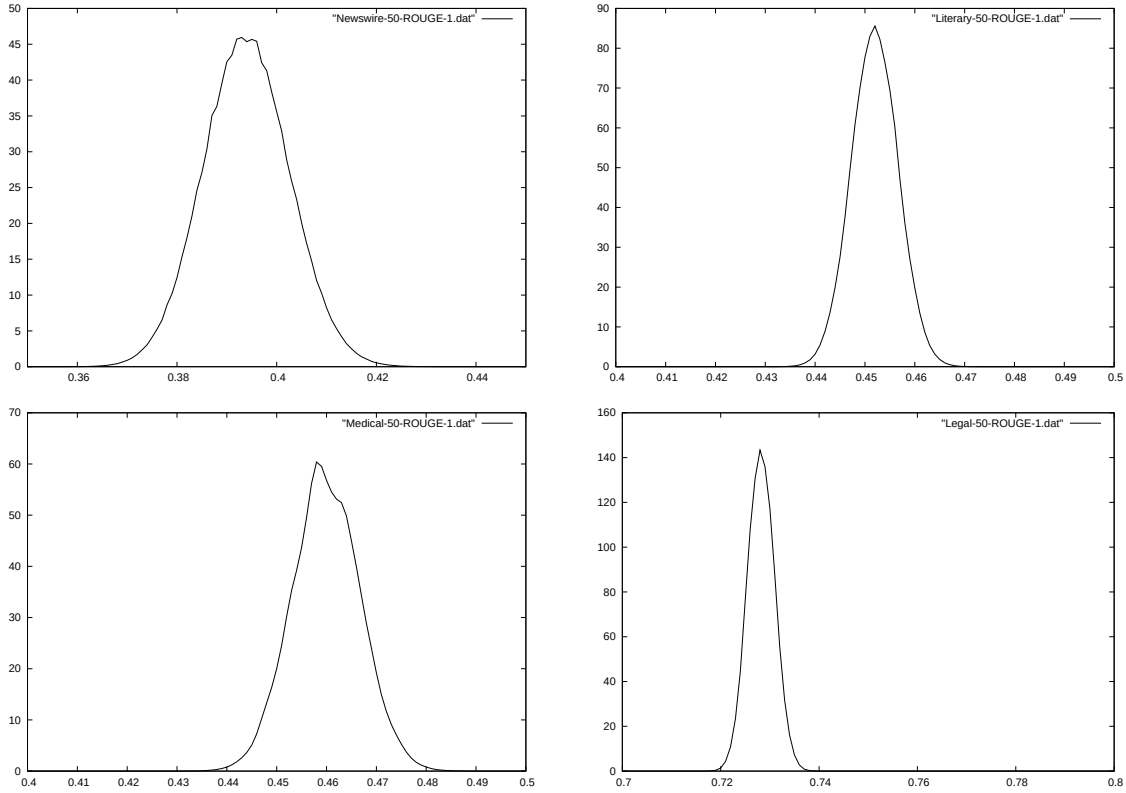

FIGURE 5.4. Probability Density Functions of ROUGE-1 recall scores for the Newswire, Literary, Scientific and Legal Domains, respectively from left to right. The resolution of the x-axis is increased to 0.1.

around this range. Furthermore, it is possible to judge the hardness of each domain by looking at the standard deviation values. A lower standard deviation indicates a steeper curve, which implies that improving a system would be harder. From the table, it can be inferred that the legal domain is the hardest while the newswire is the easiest.

Comparing Table 5.2 with the values in Table 5.1, it can be also noticed that the compression ratio affects the performance differently for each domain. For example, although the scientific domain has the highest compression ratio, it has a higher mean than the literary and the newswire domains for ROUGE-1 and ROUGE-SU4 recall scores. This implies that although the abstracts of the medical journals are highly compressed, they have a high overlap with the document, probably caused by their writing style. This was in fact confirmed earlier by the experiments in [44], where it was found out that for a data set of 188 scientific

	ROUGE-1			
Domain	μ	σ	max	min
Newswire	39.39	0.87	65.70	20.20
Literary	45.20	0.47	63.90	28.40
Scientific	45.99	0.68	71.90	24.20
Legal	72.82	0.28	82.40	62.80
	ROUGE-2			
Domain	μ	σ	max	min
Newswire	11.57	0.79	37.40	1.60
Literary	5.41	0.34	16.90	1.80
Scientific	10.98	0.60	33.30	1.30
Legal	28.74	0.29	40.90	19.60
	ROUGE-SU4			
Domain	μ	σ	max	min
Newswire	15.33	0.69	38.10	6.40
Literary	13.28	0.30	24.30	6.90
Scientific	16.13	0.50	35.80	6.20
Legal	35.63	0.25	45.70	28.70

TABLE 5.2. Statistical properties of the pdfs

articles, 79% of the sentences in the abstracts could be perfectly matched with the sentences in the corresponding documents.

Next, three different extractive summarization systems are tested on the same dataset in order to confirm the experiments. The first system implemented is called Random, and gives a random score between 1 and 100 to each sentence in a document, and then selects the top scoring sentences. The second system, Lead, implements the lead baseline method which takes the first k sentences of a document until the length limit is reached. Finally, the last system implemented is TextRank, which uses a variation of the PageRank graph centrality algorithm in order to identify the most important sentences in a document [78, 24, 68]. The reason TextRank is implemented is that it has a performance competitive with the top systems participating in DUC '02 [68]. It should also be mentioned that for the literary, scientific, and legal domains, the systems apply the algorithms for each section and each section is evaluated independently, and their resulting recall scores are summed up. This is needed in order to be consistent with the exhaustive search experiments.

	ROUGE-1		
Domain	Random	Lead	TextRank
Newswire	39.13	45.63	44.43
Literary	45.39	45.36	46.12
Scientific	45.75	47.18	49.26
Legal	73.04	72.42	74.82
	ROUGE-2		
Domain	Random	Lead	TextRank
Newswire	11.39	19.60	17.99
Literary	5.33	5.41	5.92
Scientific	10.73	12.07	12.76
Legal	28.56	28.92	31.06
	ROUGE-SU4		
Domain	Random	Lead	TextRank
Newswire	15.07	21.58	20.46
Literary	13.21	13.28	13.81
Scientific	15.92	17.12	17.85
Legal	35.41	35.55	37.64

TABLE 5.3. ROUGE recall scores of the Lead baseline, TextRank, and Random sentence selector across domains

The ROUGE recall scores of the three systems are shown in Table 5.3. As expected, for the literary and legal domains, the Random, and the Lead systems score around the mean. This is due to the fact that the leading sentences for these two domains do not indicate any significance, hence the Lead system just behaves like Random. However for the scientific and newswire domains, the leading sentences do have importance so the Lead system consistently outperforms Random. Furthermore, although TextRank is the best system for the literary, scientific, and legal domains, it gets outperformed by the Lead system on the newswire domain. This is also an expected result as none of the single-document summarization systems were able to achieve a statistically significant improvement over the lead baseline in the previous Document Understanding Conferences.

The ROUGE scoring scheme does not show how much improvement a system achieved over another, or how far it is from the upper bound. Since now it is possible to access the pdf of each domain in the data set, this information can simply be found by calculating the percentile rank of each system using the formula given in Equation 5.

	ROUGE-1		
Domain	Random	Lead	TextRank
Newswire	%39.18	%99.99	%99.99
Literary	%62.89	%62.89	%97.90
Scientific	%42.30	%95.56	%99.87
Legal	%79.47	%16.19	%99.99
	ROUGE-2		
Domain	Random	Lead	TextRank
Newswire	%39.57	%99.99	%99.99
Literary	%42.20	%54.32	%94.34
Scientific	%35.6	%96.03	%99.79
Legal	%36.68	%75.38	%99.99
	ROUGE-SU4		
Domain	Random	Lead	TextRank
Newswire	%40.68	%99.99	%99.99
Literary	%46.39	%46.39	%96.84
Scientific	%36.37	%97.69	%99.94
Legal	%23.53	%42.00	%99.99

TABLE 5.4. Percentile rankings of the Lead baseline, TextRank, and Random sentence selector across domains

The percentile ranks of all three systems for each domain are shown in Table 5.4. Notice how different the gap is between the scores of each system this time, compared to the scores in Table 5.3. For example, it can be seen from Table 5.3 that TextRank on scientific domain has only a 3.51 ROUGE-1 score improvement over a system that randomly selects sentences to include in the extract. However, Table 5.4 shows that this improvement is in fact 57.57%.

From Table 5.4, it can be seen that both TextRank and the Lead systems are in the 99.99% percentile of the newswire domain although the systems have 1.20, 1.61, and 1.12 difference in their ROUGE-1, ROUGE-2, and ROUGE-SU4 scores respectively. The high percentile for the Lead system explains why it was so hard to improve over these baseline in previous evaluations on newswire data (e.g., see the evaluations from the Document Understanding Conferences). Furthermore, Table 5.2 suggests that the upper bounds corresponding to these scores are 65.7, 37.4, and 38.1 respectively, which are well above both the TextRank and the Lead systems. Therefore, the percentile rankings of the Lead and the

TextRank systems for this domain do not seem to give any clues about how the two systems compare to each other, nor about their actual distance from the upper bounds. There are two reasons for this: First, as mentioned earlier, most of the summary space consists of easy extracts, which make the distribution long-tailed.[10] Therefore even though there are quite a bit of systems achieving high scores, their number is negligible compared to the millions of extracts that are clustered around the mean. Secondly, a higher resolution (i.e. larger number of bins) is needed in constructing the pdfs in order to be able to see the difference more clearly between the two systems. Finally, when comparing two successful systems using percentile ranks, the use of error reduction would be more beneficial.

As a final note, a random sampling of extracts from documents in the scientific and legal domains is also made, but this time without considering the section boundaries and without performing any segmentation. The number of samples for each document is kept equal to the number of extracts generated previously from the same document using a divide-and-conquer approach. The samples are evaluated using ROUGE-1 recall scores, and the pdfs are obtained for each domain using the same strategy discussed earlier in the chapter. The resulting pdfs, although exhibit similar characteristics, they have mean values (μ) around 10% lower than the ones listed in Table 5.2, which supports the findings from earlier research that segmentation is useful for text summarization.

5.9. Summary

This chapter presented a study that explores the search space of extractive summaries across four different domains. For the newswire domain all possible extracts of the given documents are generated, and for the literary, scientific, and legal domains a divide-and-conquer approach is followed by chunking the documents into sections, handling each section independently, and combining the resulting scores at the end. Then the distributions of the evaluations scores are used to generate the probability density functions (pdfs) for each domain. Various statistical properties of these pdfs helped asses the difficulty of each domain.

[10]This also accounts for the fact that even though there might be two very close ROUGE scores that are not statistically significant, their percentile rankings might differ quite a bit.

Finally, a new scoring scheme is introduced for automatic text summarization systems which can be derived from the pdfs. The new scheme calculates a percentile rank of the ROUGE-1 recall score of a system, which gives scores in the range [0-100]. This makes it possible to see how far each system is from the upper bound, and thus make a better comparison among the systems. The new scoring system showed that while there is a 20.1% gap between the upper bound and the lead baseline for the news domain, closing this gap is difficult, as the percentile rank of the lead baseline system, 99.99%, indicates that the system is already very close to the upper bound.

Furthermore, except for the literary domain, the percentile rank of the TextRank system is also very close to the upper bound. This result does not suggest that additional improvements cannot be made in these domains, but that making further improvements using only extractive summarization will be considerably difficult. Moreover, in order to see these future improvements, a higher resolution (i.e. larger number of bins) will be needed when constructing the pdfs.

In all the experiments performed, the ROUGE [50] evaluation package is used with its ROUGE-1, ROUGE-2, and ROUGE-SU4 recall scores. It should be noted that since ROUGE performs its evaluations based on the n-gram overlap between the peer and the model summaries, it does not take other summary quality metrics such as coherence and cohesion into account. However, the goal of this work was to analyze the topic-identification stage only, which concentrates on selecting the right content from the document to include in the summary, and the ROUGE scores were found to correlate well with the human judgments on assessing the content overlap of summaries.

CHAPTER 6

UNSUPERVISED EXTRACTION

6.1. Introduction

This chapter describes a set of empirical experiments performed as a first attempt to summarizing novels. The features that are prevelant on the existing summarization systems are modified so that specific attributes of long documents can be taken into account. The combination of the features are performed in an unsupervised fashion.

In the following sections, the description of the dataset will be given first. Next, the evaluation metrics that are used in the study will follow. Then using these metrics, a benchmark that provides lower and upper bounds for the systems on this dataset will be specified. Finally, after briefly describing a current state-of-the-art summarization system that has been successfully used for the summarization of short documents, the techniques that take into account the length of the documents to significantly improve the performance of this system will be given.

6.2. Dataset

50 novels are selected from the dataset described in Chapter 3, that have objective summaries available from CliffsNotes, and GradeSaver. The documents in this collection have an average length of 92,000 words, with summaries with an average length of 6,500 words (CliffsNotes) and 7,500 words (Grade Saver). Figure 6.1 plots the length of the summaries (averaged over the two manual summaries) with respect to the length of the books. As seen in the plot, most of the novels have a length of 50,000-150,000 words, with a summary of 2,000–6,000 words, corresponding to a compression rate of about 5-15%. There are also a few very long books, with more than 150,000 words, for which the summaries tend to become correspondingly longer.

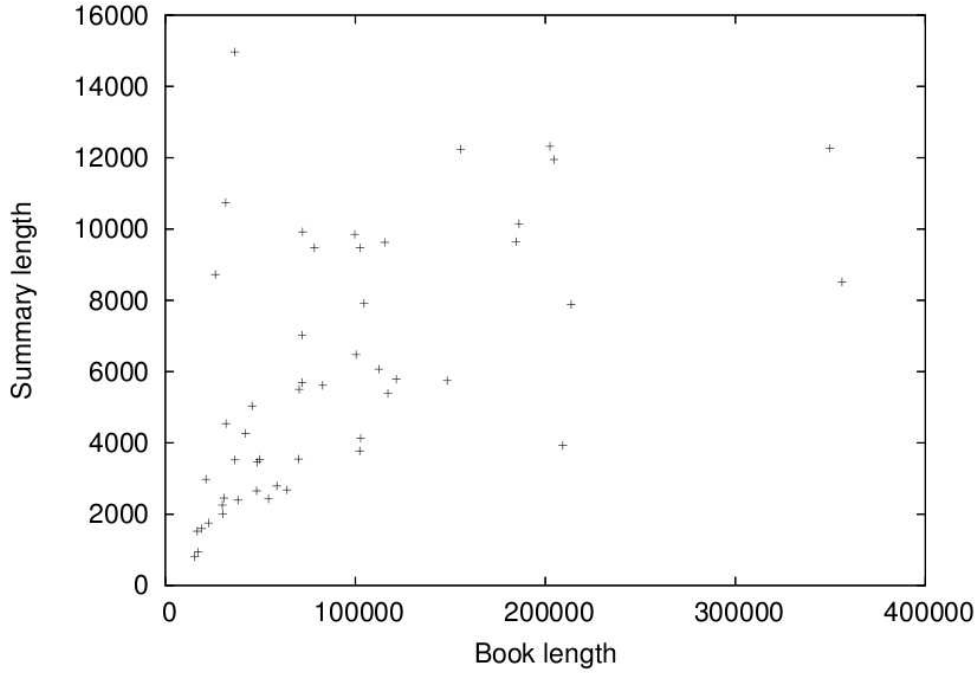

FIGURE 6.1. Summary and book lengths for 50 novels

6.2.1. Evaluation Metrics

For the evaluation, the ROUGE evaluation toolkit described in Chapter 3 is used. Throughout the chapter, the evaluations are reported using the ROUGE-1 setting, which seeks unigram matches between the generated and the reference summaries, and which was found to have high correlation with human judgments at a 95% confidence level [50] . Additionally, the final system is also evaluated using the ROUGE-2 (bigram matches) and ROUGE-SU4 (non-contiguous bigrams in addition to unigrams) settings, which have been frequently used in the Document Understanding Conference (DUC) evaluations.

In most of the previous summarization evaluations, the data sets were constructed specifically for the purpose of enabling system evaluations, and thus the length of the reference and the generated summaries was established prior to building the data set and prior to the evaluations. For instance, some of the previous DUC evaluations provided reference summaries of 100-word each, and required the participating systems to generate summaries of the same length.

However, the dataset described in the previous section consists of pre-existing summaries, with large summary-length variations across the 50 books and across the two reference summaries. To address this problem, one manual summary is kept as the main reference (GradeSaver), and other summary (CliffsNotes) is used as a way to decide on the length of the generated summaries. This means that for a given book, the CliffsNotes summary and all the automatically generated summaries have the same length, and they are all evaluated against the (possibly with a different length) Grade Saver summary.

This way, it is also possible to calculate an upper bound by comparing the two manual summaries against each other, and at the same time ensure a fair comparison between the automatically generated summaries and this upper bound.[1]

6.2.2. Lower and Upper Bounds

To determine the difficulty of the task on the 50 book data set, this section calculates and reports lower and upper bounds. The lower bound is determined by using a baseline summary constructed by including the first sentences in the novel (also known in the literature as the lead baseline).[2] As mentioned in the previous section, all the generated summaries – including this baseline – have a length equal to the CliffsNotes manual summary. The upper bound is calculated by evaluating CliffsNotes manual summary against the reference Grade Saver summary. Table 6.1 shows the precision (P), recall (R), and F-measure (F) for these lower and upper bounds, calculated as average across 50 novels.

	P	R	F
Lower bound (lead baseline)	0.380	0.284	0.325
Upper bound (manual summary)	0.569	0.493	0.528

TABLE 6.1. Lower and upper bounds for the book summarization task, calculated on the 50 book data set

[1]An alternative solution would be to determine the length of the generated summaries using a predefined compression rate (e.g., 10%). However, this again implies great variations across the lengths of the generated versus the manual summaries, which can result in large and difficult to interpret variations across the ROUGE scores.

[2]A second baseline that accounts for text segments is also calculated and reported in section 6.5.

An automatic system evaluated on this data set is therefore expected to have an F-measure higher than the lower bound of 0.325, and it is unlikely to exceed the upper bound of 0.528 obtained with a human-generated summary.

6.3. An Initial Summarization System

The first summarization experiment was done using a re-implementation of an existing state-of-the-art summarization system, which is a centroid-based method implemented in the MEAD system [82]. MEAD is chosen for three main reasons. First, MEAD was shown to lead to good performance in several DUC evaluations, e.g., [80, 47]. Second, it is an unsupervised method which, unlike supervised approaches, does not require training data. Finally, the centroid-based techniques implemented in MEAD is time wise an efficient technique, which is an important aspect in the summarization of very long documents such as novels.

The latest version of MEAD[3] uses features, classifiers and re-rankers to determine the sentences to include in the summary. The default features included are centroid, position and sentence length. The centroid value of a sentence is the sum of the centroid values of the words in the sentence. The centroid value of a word is calculated by multiplying the term frequency (tf) of a word by the word's inverse document frequency (idf) obtained from the Topic Detection and Tracking (TDT) corpus. The tf of a word is calculated by dividing the frequency of a word in a document cluster by the number of documents in the cluster. The positional value P_i of a sentence is calculated using the formula given in [82]:

$$P_i = \frac{n - i + 1}{n} * C_{max} \tag{6}$$

where n represents the number of sentences in the document, i represents the position of the sentence inside the text, and C_{max} is the score of the sentence that has the maximum centroid value.

[3]MEAD 3.11, http://www.summarization.com/mead/

	P	R	F
MEAD (original download)	0.423	0.296	0.348
MEAD (our implementation)	0.435	0.323	0.369

TABLE 6.2. Summarization results using the MEAD system

The summarizer combines these features to give a score to each sentence. The default setting consists of a linear combination of features that assigns equal weights to the centroid and the positional values, and only considers the sentences that have more than nine words. After the sentences are scored, the re-rankers are used to modify the scores of a sentence depending on its relation with other sentences. The default re-ranker implemented in MEAD first ranks the sentences by their scores in descending order and iteratively adds the top ranked sentence if the sentence is not too similar to the already added sentences. This similarity is computed as a cosine similarity and by default the sentences that exhibit a cosine similarity higher than 0.7 are not added to the summary. The sole purpose of the re-ranker in MEAD is to eliminate the similar sentences coming from different sources in multi-document summarization. Note that although the MEAD distribution also includes an optional feature calculated using the LexRank graph-based algorithm [24], this feature could not be used for novels since it takes days to compute LexRank for very long documents without any modifications, and thus its application was not tractable.

Although the MEAD system is publicly available for download, in order to be able to make continuous modifications easily and efficiently to the system as the new methods are developed, a new version is implemented. The implementation differs from the original one in certain aspects. First, the document frequency counts are determined using the British National Corpus (BNC) rather than the TDT corpus. Second, the sentence scores are normalized by dividing the score of a sentence by the length of the sentence, and instead the sentence length feature used by MEAD as a cutoff is eliminated. Note also that stop words are not taken into account when calculating the length of a sentence. Finally, a re-ranker is not used as the work in this thesis is concerned with single-document summarization rather than multi-document summarization.

Table 6.2 shows the results obtained on the dataset using the original MEAD implementation, as well as the new implementation. Although the performance of this system is clearly better than the baseline (see Table 6.1), it is nonetheless far below the upper bound. In the following section, techniques for improving the quality of the generated summaries are explored by accounting for the length of the documents.

6.4. Techniques for the Summarization of Novels

In this section, several changes are made to the initial system in order to account for the specifics of the dataset. In particular, since the dataset consists of very large documents, correspondingly the summarization of such documents requires techniques that account for their length.

6.4.1. Sentence Position in Very Large Documents

The general belief in the text summarization literature [22, 57] is that the position of sentences in a text represents one of the most important sources of information for a summarization system. In fact, a summary constructed using the lead sentences was often found to be a competitive baseline, with no systems exceeding this baseline with a statistical significance during the DUC summarization evaluations.

Although the position of sentences in a document seems like a pertinent heuristic for the summarization of short documents, the hypothesis here is that this heuristic may not hold for the summarization of very long documents such as novels, where the leading sentences do not overlap much with the essence of the document.

To test this hypothesis, the initial system is modified so that it does not account for the position of the sentences inside a document, but it only accounts for the weight of the constituent words. Correspondingly, the score of a sentence is determined only as a function of the word centroids and length, and excludes the positional score. Table 6.3 shows the average ROUGE scores obtained using the summarization system with and without the position scores.

	P	R	F
With positional scores	0.435	0.323	0.369
Without positional scores	0.459	0.329	0.383

TABLE 6.3. Summarization results with and without positional scores

As suspected, removing the position scores leads to a better overall performance, with an increase observed in both the precision and the recall of the system. Although the position in a document is a heuristic that helps the summarization of news articles and other short documents, it appears that the sentences located toward the beginning of a book are not necessarily useful for building the summary.

6.4.2. Text Segmentation

A major difference between the short and the long documents stands in the frequent topic shifts typically observed in the latter. While short stories are usually concerned with one topic at a time, long documents such as books often cover much more than a single topic. Thus, the intuition is that a summary should include content covering the important aspects of all the topics in the document, as opposed to only generic aspects relevant to the document as a whole. A system for the summarization of long documents should therefore extract key concepts from all the topics in the document, and this task is better performed when the topic boundaries are known prior to the summarization step.

To accomplish this, the system is augmented with a text segmentation module that attempts to determine the topic shifts, and correspondingly splits the document into smaller segments. Note that although the chapter boundaries are available in some of the novels in the data set, this is not always the case as there are also books in the data set for which the chapters are not explicitly identified. Furthermore, the chapters are usually coarse grained as segments, and need further chunking into fine grained segments. Therefore, to ensure a uniform treatment of the entire data set, an automatic text segmentation algorithm is applied on the novels rather than simply treating the chapters as segments.

While several text segmentation systems have been proposed to date [33, 13, 99]), a graph-based segmentation algorithm using normalized-cuts [56] is used, as it is recently shown to exceed the performance of alternative segmentation methods, and it allows the user to specify the number of segments to be created as a parameter. Very briefly, the segmentation algorithm starts by modeling the text as a graph, where sentences are represented as nodes in the graph, and inter-sentential similarities are used to draw weighted edges. The similarity between sentences is calculated using cosine similarity, with a smoothing factor that adds the counts of the words in the neighborhood sentences. Words are weighted using an adaptation of the tf.idf metric, where a document is uniformly split into chunks that are used for the tf.idf computation. There are two parameters that have to be set in this algorithm: (1) the length in words of the blocks approximating sentences; and (2) the cut-off value for drawing edges between nodes. Since the method was originally developed for spoken lecture segmentation, different parameters had to be used than suggested in [56]. Thus a development set of three books is used to optimize the parameters, and as a result the optimal sentence word-length is determined as 20, and the optimal cut-off value is determined as 25.

Once the text is divided into segments, a separate summary is generated for each segment, and consequently a final summary is created by collecting sentences from the individual segment summaries in a round-robin fashion. That is, starting with the ranked list of sentences generated by the summarization algorithm for each segment, one sentence at a time is picked from each segment summary until the desired book-summary length is reached.

A useful property of the normalized-cut segmentation algorithm is that one can decide apriori the number of segments to be generated, so it is possible to evaluate the summarization algorithm for different segmentation granularities. Figure 6.2 shows the average ROUGE-1 F-measure score obtained for summaries generated using 1 to 50 segments.

As seen in the figure, segmenting the text helps the summarization process. The average ROUGE-1 F-measure score raises to more than 0.39 F-measure for increasingly

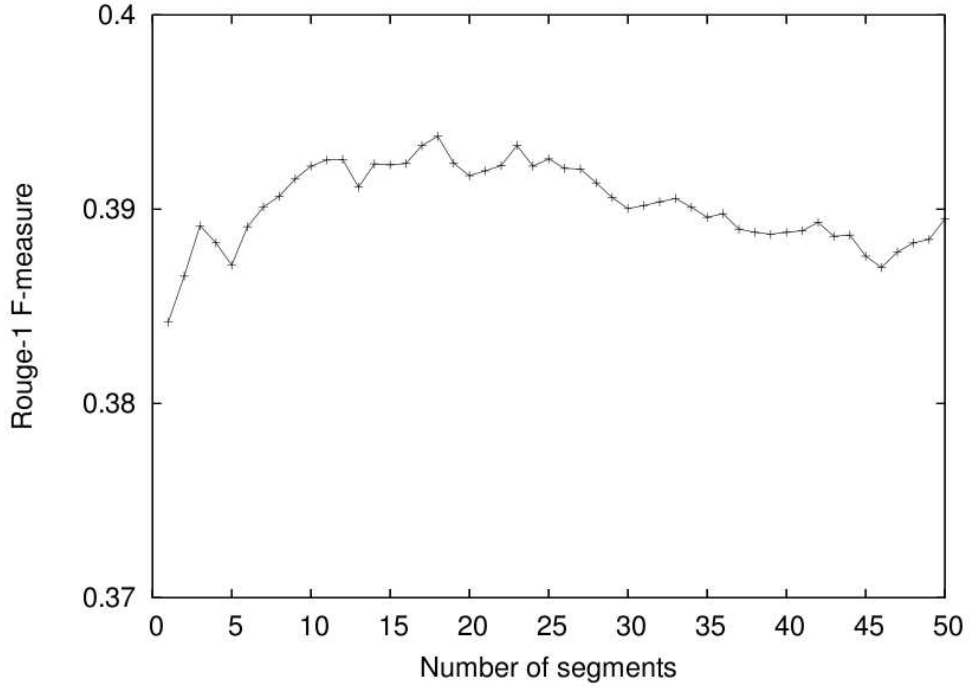

FIGURE 6.2. Summarization results for different segmentation granularities.

larger number of segments, with a plateau reached at approximately 15–25 segments, followed by a decrease when more than 30 segments are used.

In all the following evaluations, each novel is segmented into a constant number of 15 segments; in future work, more sophisticated methods should be considered for finding the optimal number of segments individually for each book.

6.4.3. Modified Term Weighting

An interesting characteristic of documents with topic shifts is that words do not have a uniform distribution across the entire document. Instead, their distribution can vary with the topic, and thus the weight of the words should change accordingly.

To account for the distribution of the words inside the entire book, as well as inside the individual topics (segments), a weighting scheme is devised that accounts for four factors: the segment term frequency (stf), calculated as the number of occurrences of a word inside a segment; the book term frequency (tf), determined as the number of occurrences of a word inside a book; the inverse segment frequency (isf), measured as the inverse of the number

of segments containing the word; and finally, the inverse document frequency (idf), which takes into account the distribution of a word in a large external corpus (as before, the BNC corpus is used). A word weight is consequently determined by multiplying the book term frequency with the segment term frequency, and the result is then multiplied with the inverse segment frequency and the inverse document frequency. This weighting scheme is referred to as tf.stf.idf.isf.

Using this weighting scheme, a word is prevented from having the same score across the entire book, and instead a higher weight is given to its occurrences in segments where the word has a high frequency. For instance, the word doctor occurs 30 times in one of the novels in the data set, which leads to a constant tf.idf score of 36.76 across the entire book. Observing that from these 30 occurrences, 19 appear in just one segment, the tf.stf.idf.isf weighting scheme will lead to a weight of 698.49 for that segment, much higher than e.g. the weight of 36 calculated for other segments that have only a few occurrences of this word.

	P	R	F
tf.idf weighting	0.463	0.339	0.391
tf.stf.idf.isf weighting	0.464	0.349	0.398

TABLE 6.4. Summarization results using a weighting scheme accounting for the distribution of words inside and across segments

In this way, the role played by a word inside a segment is emphasized, by accounting for its frequency in the segment, and discounting the words that appear in a large number of segments. Table 6.4 shows the summarization results obtained for the new weighting scheme (recall that all the results are calculated for a text segmentation into 15 segments).

6.4.4. Combining Summarization Methods

The next improvement made is to bring an additional source of knowledge into the system, by combining the summarization provided by the current system with the summarization obtained from a different method.

A variation of a centrality graph-based algorithm is implemented for unsupervised summarization, which was successfully used in the past for the summarization of short

documents. Very briefly, the TextRank system [68] – similar in spirit with the concurrently proposed LexRank method [24] – works by building a graph representation of the text, where sentences are represented as nodes, and weighted edges are drawn using inter-sentential word overlap. An eigenvector centrality algorithm is then applied on the graph (e.g., PageRank), leading to a ranking over the sentences in the document.

An impediment encountered for the algorithm was the size of the graphs, which become intractably large and dense for very large documents such as the novels in the dataset. Hence a cut-off value is used for drawing edges between the nodes, and consequently all the edges between the nodes that are farther apart than a given threshold are removed. A threshold value of 75 is empirically found to work best on the same development set of three books used to optimize the parameters for the segmentation algorithm.

	P	R	F
Our system	0.464	0.349	0.398
TextRank	0.449	0.356	0.397
COMBINED	0.464	0.363	0.407

TABLE 6.5. Summarization results for individual and combined summarization algorithms

Using the same segmentation as before (15 segments), the TextRank method by itself did not lead to improvements over the current centroid-based system. Instead, since it is noticed that the summaries generated with the current system and with TextRank covered different sentences, a method that combines the top ranked sentences from the two methods is implemented. Specifically, the combination method picks one sentence at a time from the summary generated by the current system for each segment, followed by one sentence selected from the summary generated by the TextRank method, and so on. The combination method also specifically avoids redundancy. Table 6.5 shows the results obtained with the current centroid-based system, the TextRank method, as well as the combined method.

6.4.5. Segment Ranking

In the current system, all the segments identified in a book have equal weights. However, this might not always be the case, as there are sometimes topics inside the book that have higher importance, and which consequently should be more heavily represented in the generated summaries.

To account for this intuition, a segment ranking method that assigns to each segment a score reflecting its importance inside the book is implemented. The ranking is performed with a method similar to TextRank, using a random-walk model over a graph representing segments and segment similarities. The resulting segment scores are multiplied with the sentence scores obtained from the combined method described before, normalized over each segment, resulting in a new set of scores. The top ranked sentences over the entire book are then selected for inclusion in the summary. Table 6.6 shows the summarization results obtained by using segment ranking.

	P	R	F
COMBINED	0.464	0.363	0.407
COMBINED + Segment Ranking	0.471	0.363	0.410

TABLE 6.6. Summarization results using segment ranking

6.5. Discussion

In addition to the ROUGE-1 metric, the quality of the summaries generated with the final summarization system was also evaluated using the ROUGE-2 and ROUGE-SU4 metrics, which are frequently used in the DUC evaluations. Table 6.7 shows the figures obtained with ROUGE-1, ROUGE-2 and ROUGE-SU4 for the final system, for the original MEAD download, as well as for the lower and upper bounds. The table also shows an additional baseline determined by selecting the first sentences in each segment, using the segmentation into 15 segments as determined before.

Regardless of the evaluation metric used, the performance of the novel summarization system is significantly higher than the one of an existing state-of-the-art summarization

	ROUGE-1			ROUGE-2			ROUGE-SU4		
	P	R	F	P	R	F	P	R	F
Lead Baseline	0.380	0.284	0.325	0.035	0.037	0.036	0.096	0.073	0.083
Segment baseline	0.402	0.301	0.344	0.040	0.031	0.035	0.102	0.079	0.089
MEAD	0.423	0.296	0.348	0.039	0.029	0.033	0.106	0.076	0.088
The System	0.471	0.363	0.410	0.069	0.054	0.061	0.148	0.114	0.129
Upper Bound	0.569	0.493	0.528	0.112	0.097	0.104	0.210	0.182	0.195

TABLE 6.7. Evaluation of the final summarization system using different ROUGE metrics

system that has been designed for the summarization of short documents (MEAD). In fact, if the upper bound of 0.528 is accounted, the relative error rate reduction for the ROUGE-1 F-measure score obtained by the system with respect to MEAD is a significant 34.44%.

The performance of the system is mainly due to the features that account for the length of the document: exclusion of positional scores, text segmentation and segment ranking, and a segment-based weighting scheme. An additional improvement is obtained by combining two different summarization methods. It is also worth noting that the final system is quite efficient, taking about 200 seconds to apply the segmentation algorithm, plus an additional 65 seconds to generate the summary of one novel.[4]

To assess the usefulness of the final system with respect to the length of the documents, the individual results obtained for books of different sizes are analyzed. Averaging the results obtained for the shorter novels in the collection, i.e. 17 books that have a length between 20,000 and 50,000 words, the lead baseline gives a ROUGE-1 F-measure score of 0.337, the proposed system leads to 0.378, and the upper bound is measured at 0.498, indicating a relative error rate reduction of 25.46% obtained by the system with respect to the lead baseline (accounting for the maximum achievable score given by the upper bound). Instead, when only the books with a length over 100,000 words are considered (16 books in the data set fall under this category), the lead baseline is determined as 0.347, the proposed system leads to 0.418, and the upper bound is calculated as 0.552, which results in a higher 34.64% relative error rate reduction. This suggests that the proposed system is even more

[4]Running times measured on a Pentium IV 3GHz, 2GB RAM.

effective for longer books, due perhaps to the features that specifically take into account the length of the books.

There are also cases where the system does not improve over the baseline. For instance, for the summarization of Candide by François Voltaire, the system achieves a ROUGE-1 F-measure of 0.361, which is slightly worse than the lead baseline of 0.368. There are also cases where the performance of the system comes close to the upper bound, as it is the case with the summarization of The House of the Seven Gables by Nathaniel Hawthorne, which has a lead baseline of 0.296, an upper bound of 0.457, and the proposed system obtains 0.404. This either indicates that the summarization styles differ for some of the books in the dataset, or perhaps a possible avenue for future research is to account for the characteristics of a book, and devise summarization methods that can adapt to the specifics of a given book such as length, genre, and others.

6.6. Related Work

There are two main trends that can be identified in the summarization literature: supervised systems, that rely on machine learning algorithms trained on pre-existing document-summary pairs, and unsupervised techniques, based on properties and heuristics derived from the text.

Among the unsupervised techniques, typical summarization methods account for both the weight of the words in sentences, as well as the sentence position inside a document. One of the most successful systems is perhaps the centroid approach [82], which extends the idea of tf.idf weighting [87] by introducing word centroids, as well as integrating other features such as position, first-sentence overlap and sentence length. More recently, graph-based methods that rely on sentence connectivity have also been found successful, using algorithms such as node degree [87] or eigenvector centrality [68, 24, 102].

In addition to unsupervised methods, supervised machine learning techniques have also been used with considerable success. Assuming the availability of a collection of documents and their corresponding manually constructed summaries, these methods attempt to identify the key properties of a good summary, such as the presence of named entities,

positional scores, or the location of key phrases. Such supervised techniques have been successfully used in the systems proposed by e.g. [97, 34, 108, 20].

In addition to short news documents, which have been the focus of most of the summarization systems proposed to date, only relatively little work has been carried out on the summarization of other types of documents. This includes systems addressing the summarization of e-mail threads [100], online discussions [109], or spoken dialogue [28].

6.7. Summary

In this chapter, an unsupervised system that attempts to summarize novels is given. The system is composed of individual features take take into account the characteristics of long documents. With an ad-hoc combination of features, the system significantly outperforms MEAD, a state-of-the-art system developed for short documents.

The work makes two important contributions. First, it introduced a new summarization benchmark, specifically targeting the evaluation of systems for the summarization of literary novels. Second, it showed that state-of-the-art summarization systems developed for the summarization of short documents do not fare well when applied to very long documents, and instead a better performance can be achieved with a system that accounts for the length of the documents. In particular, the summarization system that is developed was found to lead to more than 30% relative error rate reduction with respect to an existing state-of-the-art summarization tool.

CHAPTER 7

SUPERVISED EXTRACTION

7.1. Introduction

This chapter is concerned with the process of building a supervised extractive summarization system. Using the results from the exhaustive evaluations presented in Chapter 5, the system learns how to generate good summaries by optimizing a sentence ranking function that makes use of various features, which are basically unsupervised extractive summarization systems. The evaluation results show that a supervised approach to feature combination generates better summaries in two domains than the unsupervised systems discussed so far. In order to confirm the intuition from the previous chapter that the success of the features depend on the genre, two different domains are analyzed; news, and literary novels.

The previous chapter revealed the fact that the traditional features do not suit well to literary novels, hence they need adjustments, or even new features should be introduced in order to build successful summarization systems in this domain. With this guidance, a new system was introduced that uses a combination of such features, which in turn outperformed the traditional summarization systems designed for news domain. However the new system combines the mentioned features in an ad-hoc fashion, based on empirical observation. For example, the final system in the previous chapter, combined the TextRank feature and the new frequency based feature by giving equal weights to both. This chapter formalizes this combination process by making use of machine learning algorithms that learns the weights from training sets generated by using the exhaustive evaluations performed in Chapter 5.

In the remaining of this chapter; first, a description of the dataset will be given, followed by the description, and the evaluation of the individual features used. Then the experimental setup section will explain how the training instances are generated, hence how the training and test sets are developed. Next, the evaluation results of the experiments will

Type	N	μ_{Dw}	μ_{Sw}	μ_R
Training	197	475	101	79%
Test	50	655	101	85%

TABLE 7.1. Statistical properties of the news data. N represents the number of documents; μ_{Dw}, and μ_{Sw} represent the average number of words for each document and summary respectively; and μ_R indicates the average compression ratio.

be presented and discussed. Finally, the chapter will conclude with related work in this area, and overall summary of the chapter.

7.2. Dataset

In order to confirm the intuitions from the previous chapter that different features would be prevalent in different domains, the dataset is constructed from two different domains; news and literary novels.

247 documents are collected from the news domain, where 197 of them are used for training and 50 for testing. For the news domain, the articles and their summaries are selected from the Document Understanding Conference (DUC) 2002 data set[1]. Table 7.1 shows the properties of this dataset.

The dataset selection process for the literary domain is a little more complicated. Recall from Chapter 5 that exhaustive evaluations on long documents is not tractable due to the exponentially growing search space with increasing number of sentences in a document. Therefore the novels are segmented into smaller sections in order to perform the exhaustive evaluations independently on each section, and combine the results later on. The segmentation process is done in two levels: The first level uses the chapters in the novel, defined by the authors at the time of its writing. The second level of the process uses a segmentation algorithm to further segment the chapters into smaller sections automatically. This step is needed as the table in Chapter 3 shows that each book has 108,800 words, and 32 chapters on average, which makes the length of each chapter 3,400 words on average, which is still much

[1]http://www-nlpir.nist.gov/projects/duc/data.html

Type	N	μ_{Dw}	μ_{Sw}	μ_R	μ_C	μ_{Cw}
Training	26	123,460	7430	94%	30	4115
Test	10	112,702	7220	94%	27	4174

TABLE 7.2. Statistical properties of the literary novels data. N represents the number of novels; μ_{Dw}, and μ_{Sw} represent the average number of words for each document and summary respectively; μ_R indicates the average compression ratio; and μ_C and μ_{Cw} represent the average number of sections for each document, and the average number of words for each section respectively.

longer than an average news document shown in Table 7.1. The segmentation algorithm used at this level is the publicly available linear text segmentation algorithm given in [99].[2]

For the training set in this domain, 26 novels are drawn from the collection described in Chapter 3 having a total of 773 chapters, and 4650 segments. For the test set 10 novels are selected, having a total of 270 chapters, and 1575 segments.Table 7.2 shows the properties of this dataset.

7.3. Features

The features used in training a machine learning algorithm are unsupervised extractive summarization algorithms. As mentioned throughout the thesis, the extractive summarization algorithms for single document summarization assign scores to the sentences of a given document, and select the top scoring sentences based on a given compression ratio. Thus the job of these algorithms is essentially to rank the sentences of a given document based on certain criteria. Each feature described below uses a different criterion to form its ranking function.

7.3.1. Length

This feature simply scores the sentences in a document based on their length. Hence, given a document, the ranking function simply sorts the sentences in descending order of

[2]http://mastarpj.nict.go.jp/ mutiyama/software/textseg/textseg-1.211.tar.gz

their length. In case two sentences have the same length, the selection of the higher ranked one is random.[3]

7.3.2. Position

This feature simply ranks each sentence based on its occurrence in the text. Thus the first sentence in the document gets the highest score. For a document with n sentences, the positional score P_{S_i} of sentence S_i can be given as:

$$P_{S_i} = \frac{n - i + 1}{n} \tag{7}$$

7.3.3. Term Frequency (TF)

This feature computes the number of occurrences of each term in the text by excluding the terms from a predefined list of stop words. Then each sentence in the document is assigned a score by taking the sum of the frequencies of the terms in the sentence. Formally, let sentence $S_i = w_{i1}w_{i2}...w_{im}$ be a sequence of m terms, and let the function $f(w)$ return the frequency of term w in the text. Then the term frequency score TF_{S_i} for sentence S_i can be given as:

$$TF_{S_i} = \sum_{j=1}^{m} f(w_{ij}) \tag{8}$$

Furthermore, this general formula is modified for literary novels. As mentioned in Section 7.2, as well as in Chapter 6 that since the novels are segmented, the segment frequencies of the terms should also be taken into account in the computation of this feature for this domain. Thus the straightforward application of the term frequency feature is modified by multiplying the frequency of a term in the entire document with its frequency in the segment. Formally, let $s_k(w)$ represent the frequency of term w in the k^{th} segment. Then the term frequency of sentence S_{ik} in segment k is given as:

[3]In this, and the other features described below, it should be kept in mind that only the functional words, i.e. the words that are not stop words are considered.

(9) $$TF_{S_{ik}} = \sum_{j=1}^{m} f(w_{ij}) \times s_k(w_{ij})$$

7.3.4. Term Frequency * Inverse Document Frequency (TF*IDF)

The term frequency feature is often discounted with inverse document frequency in order to diminish the weight of the terms occurring very often in the corpus. The inverse document frequency gives a term's general importance in the corpus. For example, in a corpus of computer science articles, the word computer may be very frequent yet not too important for any of the documents, as it occurs in almost all of them, i.e. it has a high document frequency. Hence the term's importance should be offset by the number of documents in the corpus in which the term is found. Formally, let D represent the set of documents in the corpus, w represent a term, and $d \in D$, then the inverse document frequency of a term w is defined as:

(10) $$idf(w) = log \frac{|D|}{|d : w \in d|}$$

Therefore the combined metric, TFIDF [86], of a term w is simply given as $TFIDF(w) = f(w) \times idf(w)$. It should be noted that slight variations of the TFIDF formula exist in literature. Furthermore, the TFIDF score of a sentence $S_i = w_{i1}w_{i2}...w_{im}$ is assigned the same way as the TF scores are assigned, i.e. by summation of the weights of the individual terms of the sentence. Hence the formula is given as:

(11) $$TFIDF_{S_i} = \sum_{j=1}^{m} TFIDF(w_{ij})$$

The above formula is also modified for literary novels in order to take segment level frequencies into account. The additional feature used here is the inverse segment frequency (isf), which is also introduced in Chapter 6. Similar to the idea of inverse document frequency,

the inverse segment frequency of a term is calculated from the number of segments containing the term. Formally, let $L = l_1, l_2, ..., l_k$ denote the set of segments in the document, and $l \in L$ denote a segment, then the inverse segment frequency of a term w is given as:

$$isf(w) = log\frac{|L|}{|l : w \in l|} \tag{12}$$

Hence the combined TFIDF value of a term w for a term in segment k is defined as $TFIDF_k(w) = f(w) \times s_k(w) \times isf(w) \times idf(w)$. Thus the feature for the sentence S_ik in segment k can be computed as:

$$TFIDF_{S_{ik}} = \sum_{j=1}^{m} TFIDF_k(w_{ij}) \tag{13}$$

7.3.5. Named Entities

This feature is similar to the Term Frequency but the terms in this case are formed from the named entities instead of words. Thus, if we let E represent the set of named entities in the document, and reuse the function $f()$ to compute the frequency of a named entity, the feature can be formalized as:

$$NE_{S_i} = \sum_{e \in E} f(e) \tag{14}$$

The detection of the named entities in the document is performed by using a publicly available implementation as part of the Natural Language Processing Toolkit [7].

7.3.6. Code Quantity Principle

This feature is a model of a linguistic theory of the same name. In short, the Code Quantity Principle states that if the part of the text that carries more important, or less predictable, or simply just more information compared to another part, then it will also be longer. This is in fact in line with other measures in information theory such as the

descriptive complexity, and the Shannon Entropy. Simply put, the more you can compress the information, the less important it is and the less amount of code it requires for its representation.

A simple model built around this theory that is used here is given in [54]. In order to assign a score to the importance of a sentence, the model assumes that the information is contained in the noun phrases, and the longer the noun phrase is, the more information it contains. Hence, the scoring function simply counts the number of words belonging to noun phrases in the sentence, and then normalizes this count by the number of noun phrases in the sentences. Formally, let P denote the set of phrases in sentence S_i, and $r(p)$ be the function that returns the number of function words in phrase p, then the score of sentence S_i is assigned by:

$$CQP_i = \frac{\sum_{p \in P} r(p)}{|P|} \tag{15}$$

It should be noted that even though the equation has a normalization factor, i.e. the number of phrases in the sentence, there is nothing stopping the model from preferring longer phrases.

7.3.7. Lexical Chains

This feature relies on the cohesive structure of the text modeled by lexical chains. The notion of cohesion is first introduced in [31], and is defined as the links that hold the different parts of the text together. Among five categories of devices introduced in [31] that are used to build a cohesive text, lexical cohesion is perhaps the easiest to model as it can be identified by considering only the surface level forms of the text. Lexical cohesion occurring among sequences of related words is referred to as lexical chains [71].

The use of lexical chains as a source representation for text summarization is first introduced in [3]. The lexical chains are used together with a text segmentation algorithm due to the close connection between the discourse structure and the cohesive structure of the text. The objective is to reveal the central theme or discourse unit in the text, which in turn

can be used for summarization. The feature used here relies entirely on the idea described in this paper.

In summary, the algorithm described in [3] identifies the semantically related nouns or noun phrases in each segment of the text, and forms a relation graph between them based on their distance in the text as well as their semantic relations identified through WordNet [26]. Every possible relation in the semantic space is considered in forming the relation graph, and the best interpretation is selected for each segment at the end. Then the chains in each interpretation across all the segments are combined, and the final chains are scored based on the links formed between semantically related words. The top scoring chains above a certain threshold picked automatically from the distribution are identified as strong chains. The sentences forming the summary are then selected based on the representative words in the strong chains. One sentence is picked for each strong chain for which a representative word makes its first occurrence.

It should be noted however that this algorithm does not give a well defined ranking for all of the sentences in the document as it relies only on the strong chains in selecting the sentences forming the summary. As it will be shown in Section 7.5, a complete ranking of the sentences are needed in order to generate the training instances in the framework presented in this chapter. Therefore the lexical chains algorithm is modified such that as many sentences as possible are selected for the strong chains first, and then the same process is applied for weak chains. If there are any remaining sentences, then they are randomly ordered. The sentence selection from a chain is essentially the same except that one sentence is selected for each representative word in the chain.

7.3.8. TextRank

This feature is a graph-based sentence ranking algorithm that tries to capture the centrality of a document by taking into account the relations between the sentences. It is a variation of the PageRank Algorithm [78], which is used by the search engines to rank web documents. The TextRank algorithm is similar in spirit to LexRank [24], and in general it can be applied for various text processing tasks that require a ranking of the individual text units

such as keyword extraction. In Chapter 6, the algorithm is used both for summarization, and segment ranking tasks. This feature uses the algorithm for summarization, i.e. to rank the sentences of a given document.

Briefly, the TextRank algorithm constructs a graph representation of the text, where sentences are represented as nodes, and weighted edges are drawn using inter-sentential word overlap. An eigenvector centrality algorithm is then applied on the graph, leading to a ranking over the sentences in the document. Formally, let $S(V_i)$ denote the value of the node V_i, $In(V_i)$ denote the set of nodes that point to V_i, and $Out(V_i)$ denote the set of nodes that V_i points to. Then the equation to calculate the value of a single node can be given as:

$$S(V_i) = (1-d) + d \times \sum_{V_j \in In(V_i)} \frac{w_j i}{\sum_{V_k \in Out(V_j)} w_j k} S(V_j) \tag{16}$$

The algorithm starts by assigning constant values to each node, and converges after the change in values after a single iteration is below a certain threshold. In this work, this change is calculated as the square root of the sum of the squares of the differences of node values between successive iterations. Formally, let V represent the set of nodes in the graph, $S_m(V_i)$ be the value of the node V_i in iteration m, then the formula to calculate the change C_m can be given as:

$$C_m = \sqrt{\sum_{V_i \in V} (S_m(V_i) - S_{m-1}(V_i))^2} \tag{17}$$

Once the algorithm converges each node value represents the score of the sentence. Hence the ranking is performed by sorting the sentences in descending order of their TextRank scores.

7.4. Evaluation of the Features

Each feature described in the previous section give a different ranking for the sentences of a document. Moreover, the top ranked sentences from those rankings are selected in order

Feature	ROUGE-1	ROUGE-2	ROUGE-SU4
Length	0.406	0.137	0.170
Position	0.468	0.204	0.212
TF	0.428	0.161	0.189
TF / Length	0.450	0.178	0.205
TF * IDF	0.425	0.158	0.186
TF * IDF / Length	0.450	0.178	0.205
Named Entities	0.405	0.135	0.169
Named Entities / Length	0.422	0.152	0.182
Code Quantity Principle	0.410	0.152	0.178
Lexical Chains	0.454	0.186	0.209
TextRank	0.444	0.172	0.201

TABLE 7.3. The ROUGE Recall results of the evaluation of each individual feature on news domain.

to obtain the extractive summary. The number of top ranked sentences to include in the summary depend on the number of words in the model summary. The Table 7.3, and Table 7.4 present the ROUGE evaluation results obtained for the news and novels corpora respectively.

Note that variations of some of the features that are normalized by the sentence length are also considered. These variations are formed by combination of each feature with the length feature, and is simply obtained by dividing the feature's score by the number of functional terms in the sentence.

7.5. Generating Training Instances

If the evaluation results for the individual features given in the previous section are used directly to train a machine learning algorithm together with the exhaustive evaluation results from Chapter 5, then the final system will learn how to pick the best summary out of a given set of summaries, i.e. it will not learn how to select the sentences to include in the summary but rather select the best summary generated out of a given set. Hence to test this framework, all the possible summaries for a given document should be exhaustively generated. This however is not feasible, and does not provide any benefit due to the obvious fact that if one can generate all the possible summaries, then these summaries can also be

Feature	ROUGE-1	ROUGE-2	ROUGE-SU4
Length	0.423	0.078	0.156
Position	0.419	0.073	0.149
TF	0.443	0.085	0.164
TF / Length	0.441	0.079	0.157
TF * IDF	0.457	0.088	0.171
TF * IDF / Length	0.452	0.082	0.162
Named Entities	0.447	0.075	0.159
Named Entities / Length	0.444	0.074	0.158
Code Quantity Principle	0.426	0.073	0.151
Lexical Chains	0.409	0.067	0.142
TextRank	0.454	0.089	0.173

TABLE 7.4. The ROUGE Recall results of the evaluation of each individual feature on literary novels domain.

evaluated at the same time, and the top one can easily be picked. Therefore, the supervised system designed for this step should learn how to optimize the sentence ranking process for a given document by looking at how each feature ranks them, and how the rankings of the features overlap with the results from the exhaustive evaluations.

With this idea in mind, training instances are generated per sentence of every document in the training data. For each sentence, the value of a feature is assigned by considering the rank of the sentence; assigning higher values to sentences that are ranked higher by the feature. Hence for a given sentence S_i, with a ranking i given by a feature f, its value is defined by the function $g(S_i)$ as:

$$g(S_i) = \frac{1}{\log i} \tag{18}$$

The label values are assigned by using the exhaustive evaluations from Chapter 5. For a given sentence S, the average of the recall scores of all the possible summaries the sentence is included is calculated. It should be noted that this is not an ideal labeling situation but only an approximation. The assumption here is that if a sentence is picked in a summary, then the summary's score should reflect whether the choice of that sentence in

Feature	ROUGE-1	ROUGE-2	ROUGE-SU4
Linear Regression	0.473	0.207	0.228
Multi-layer Perceptron	0.476	0.211	0.232
SMOReg	0.476	0.212	0.232

TABLE 7.5. The ROUGE Recall results of the evaluation of machined learned summarizers on news domain.

the summary is a good choice or not. The assumption might fail for certain individual cases where the sentence could be a perfect candidate to be included in the summary but due to the existence of other sentences, the overall score of the summary can be low. However since all the possible scores are averaged, and the Chapter 5 showed that the extractive summary space follow a Gaussian Distribution, the quality of a sentence would be expected to be reflected in the label score.

Thus given M sentences, and K features, M training instances of the form (x^i, y^i) for $i = 1, 2, ..., M$, where x is a K dimensional vector, and y represents the labels, is generated. Three different supervised learning algorithms are then trained on this data using the Weka Toolkit [30]. The supervised algorithms used are; linear regression, multi-layer perceptron, and SMOReg, a sequential minimal optimization algorithm for training a support vector regression using polynomial or RBF kernels, presented in [91].

7.6. Results and Discussions

Once the supervised learning algorithms are trained, they are tested on the test data. However, since the algorithms learn a ranking function, the resulting system assigns a real valued score to each sentence. Thus the results do not immediately tell us how the new system performs. The sentences still need to be ranked in descending order of their scores assigned by the new system, and the top sentences need to be picked according to the compression ratio, determined by the number of words in the model summary. Once the top sentences are picked, they are fed as input to ROUGE for the final evaluation results. Table 7.5, and Table 7.6 lists these results for the three supervised learning methods discussed in the previous section for the news and novels domain respectively.

Feature	ROUGE-1	ROUGE-2	ROUGE-SU4
Linear Regression	0.459	0.089	0.171
Multi-layer Perceptron	0.455	0.083	0.168
SMOReg	0.462	0.090	0.173

TABLE 7.6. The ROUGE Recall results of the evaluation of machined learned summarizers on novels domain.

For the news domain, all three supervised systems listed in Table 7.5 give superior results to each one of the individual features listed in Table 7.3. For novels, except the system that trained on the multi layer perceptron, the systems were able to beat the TFIDF feature, which is the best scoring unsupervised system. However the multi layer perceptron still achieves a good score, beating the rest of the individual features. This shows that the framework proposed in this chapter can in fact be used to build systems that can be trained to learn a new ranking function that combines each individual feature better than the ad-hoc combinations the unsupervised systems provide.

7.7. Related Work

One of the first supervised systems in text summarization is described in [44]. The system uses a Naive-Bayes Classifier where each sentence in the source document goes through a binary classification, i.e. it is either included in the summary or not. The features used in this classifier were mainly the ones in [22] with a few new ones such as the length of the sentence, as described in Section 7.3.1. Based on a manual matching of the medical abstracts to source texts, the study found that the most successful systems are the ones that uses position, cue phrases, and sentence length features together. In [97], the same classifier is used on a different dataset, which consisted of scientific articles for which the summaries are written by the authors of the articles themselves, rather than the professional abstractors. Another study that extends the idea of Naive-Bayes classification is given in [1] which considered richer features such as word collocations. Furthermore, an attempt is also made in [1] to bring some coherence into the summaries by performing a shallow discourse analysis of the source text, and by using coreference resolution.

Instead of a Naive-Bayes Classifier, a decision tree is used in [49], which combined even more features. The features used in the study ranged from very simple ones such as whether a sentence has proper names, or numerical data, to more complex ones such as measures of lexical connectivity. However, even these more complex features were still based on the shallow statistical techniques. An important finding of the study was that even a simple ad-hoc combination of the features did as well as the decision tree classifier most of the times, and sometimes even exceeded it. Another important aspect of this study was disregarding the assumption of feature independence. In a later work, this notion is also considered via a maximum entropy model [75].

The work of [89] describes a fast query based multi-document summarization system that only uses simple term frequency based features. The learning setting presented is similar to the work described in this thesis except that the training instances are labeled using the word overlap between the sentence and the model summaries instead of exhaustive evaluations used in this study. A more complicated framework that takes the interdependencies between the sentences into account using Conditional Random Fields is presented in [90]. In [105], an attempt is made to learn to predict the appearence of individual terms in the references, independent from the sentence selection procedure. Finally, the work in [46] presents a system that jointly learns to optimize diversity, coverage, and balance. Hence the proposed system seeks to minimize redundancy by including the main points of the summary from as many aspects as possible.

7.8. Summary

In this chapter, a description of a new framework is given to build supervised extractive summarization systems using the unsupervised extraction systems as features, and results from the exhaustive evaluations from Chapter 5 as labels. The individual performance of each feature is analyzed first. Then the design process for generating training data for the supervised systems is described, followed by the application of this process on two different domains; news and literary novels. The results showed that most of the supervised

systems built in this framework give better results than each of the individual unsupervised system.

CHAPTER 8

SENTENCE COMPRESSION

8.1. Introduction

This chapter describes a simple sentence compression algorithm that relies on external knowledge. It uses the concept of self-information to judge whether adverbial phrases should be excluded from a sentence or not. In order to compute the probabilities for self-information of a given phrase, the algorithm makes heavy use of the Web1T Corpus. Thus in order to make its job easier, a fast indexing and retrieval tool for large N-gram corpus is implemented, which can also be used in many other areas of Natural Language Processing.

The newly developed tool indexes the entire Web1T dataset with an index size of only 100 MB and performs a retrieval of any N-gram with a single disk access. With an increased index size of 420 MB and duplicate data, it also allows users to issue wild card queries provided that the wild cards in the query are contiguous. Furthermore, it also implements some of the smoothing algorithms that are designed specifically for large datasets and are shown to yield better language models than the traditional ones on the Web1T 5-gram corpus [107].

A description of the indexing tool will be given first, followed by the description of the sentence compression algorithm. Then the results of applying the algorithm on the test dataset described in previous chapter will be presented, followed by a discussion of the results. The chapter will conclude with a brief description of the related work, and summary.

8.2. N-gram Indexer

The goal of statistical language modeling is to capture the properties of a language through a probability distribution so that the probabilities of word sequences can be estimated. Since the probability distribution is built from a corpus of the language by computing the frequencies of the N-grams found in the corpus, the data sparsity is always an issue with

the language models. Hence, as it is the case with many statistical models used in Natural Language Processing (NLP), the models give a much better performance with larger data sets.

However the large data sets, such as the Web1T 5-Gram corpus of [9], present a major challenge. The language models built from these sets cannot fit in memory, hence efficient accessing of the N-gram frequencies becomes an issue. Trivial methods such as linear or binary search over the entire dataset in order to access a single N-gram prove inefficient, as even a binary search over a single file of 10,000,000 records, which is the case of the Web1T corpus, requires in the worst case $\lceil log_2(10,000,000) \rceil = 24$ accesses to the disk drive.

Since the access to N-grams is costly for these large data sets, the implementation of further improvements such as smoothing algorithms becomes impractical. This chapter overcomes this problem by implementing a novel, publicly available tool[1] that employs an indexing strategy that reduces the access time to any N-gram in the Web1T corpus to a single disk access. We also make a second contribution by implementing some of the smoothing models that take into account the size of the dataset, and are shown to yield up to 31% perplexity reduction on the Brown corpus [107]. Our implementation is space efficient, and provides a fast access to both the N-gram frequencies, as well as their smoothed probabilities.

8.2.1. The Web1T 5-gram Corpus

The Web1T 5-gram corpus [9] consists of sequences of words (N-grams) and their associated counts extracted from a Web corpus of approximately one trillion words. The length of each sequence, N, ranges from 1 to 5, and the size of the entire corpus is approximately 88GB (25GB in compressed form). The unigrams form the vocabulary of the corpus and are stored in a single file which includes around 13 million tokens and their associated counts. The remaining N-grams are stored separately across multiple files in lexicographic order. For example, there are 977,069,902 distinct trigrams in the dataset, and they are stored consecutively in 98 files in lexicographic order. Furthermore, each N-gram file contains 10,000,000 N-grams except the last one, which contains less. It is also important to

[1]Our tool can be freely downloaded from the download section under http://lit.csci.unt.edu

note that N-grams with counts less than 40 are excluded from the dataset for $N = 2, 3, 4, 5$, and the tokens with less than 200 are excluded from the unigrams.

8.2.2. B$^+$-trees

We used a B$^+$-tree structure for indexing. A B$^+$-tree is essentially a balanced search tree where each node has several children. Indexing large files using B$^+$ trees is a popular technique implemented by most database systems today as the underlying structure for efficient range queries. Although many variations of B$^+$-trees exist, we use the definition for primary indexing given in [88]. Therefore we assume that the data, which is composed of records, is only stored in the leaves of the tree and the internal nodes store only the keys.

The data in the leaves of a B$^+$-tree is grouped into buckets, where the size of a bucket is determined by a bucket factor parameter, bkfr. Therefore at any given time, each bucket can hold a number of records in the range $[1, bkfr]$. Similarly, the number of keys that each internal node can hold is determined by the order parameter, v. By definition, each internal node except the root can have any number of keys in the range $[v, 2v]$, and the root must have at least one key. Finally, an internal node with k keys has $k + 1$ children.

8.2.3. Mapping Unigrams to Integer Keys

A key in a B$^+$-tree is a lookup value for a record, and a record in our case is an N-gram together with its count. Therefore each line of an N-gram file in the Web1T dataset makes up a record. Since each N-gram is distinct, it is possible to use the N-gram itself as a key. However in order to reduce the storage requirements and make the comparisons faster during a lookup, we map each unigram to an integer, and form the keys of the records using the integer values instead of the tokens themselves.[2]

To map unigrams to integers, we use the unigrams sorted in lexicographic order and assign an integer value to each unigram starting from 1. In other words, if we let the m-tuple $U = (t_1, t_2, ..., t_m)$ represent all the unigrams sorted in lexicographic order, then for a unigram t_i, i gives its key value. The key of trigram "t_i t_j t_k" is simply given as "i j k."

[2]This method does not give optimal storage, for which one should implement a compression Huffman coding scheme.

Thus, the comparison of two keys can be done in a similar fashion to the comparison of two N-grams; we first compare the first integer of each key, and in case of equality, we compare the second integers, and so on. We stop the comparison as soon as an inequality is found. If all the comparisons result in equality then the two keys (N-grams) are equal.

8.2.4. Searching for a Record

We construct a B^+-tree for each N-gram file in the dataset for $N = 2, 3, 4, 5$, and keep the key of the first N-gram for each file in memory. When a query q is issued, we first find the file that contains q by comparing the key of q to the keys in memory. Since this is an in-memory operation, it can be simply done by performing a binary search. Once the correct file is found, we then search the B^+-tree constructed for that file for the N-gram q by using its key.

As is the case with any binary search tree, a search in a B^+-tree starts at the root level and ends in the leaves. If we let r_i and p_j represent a key and a pointer to the child of an internal node respectively, for $i = 1, 2, ..., k$ and $j = 1, 2, ..., k + 1$, then to search an internal node, including the root, for a key q, we first find the key r_m that satisfies one of the following:

- $(q < r_m) \wedge (m = 1)$
- $(r_{m-1} \leq q) \wedge (r_m > q)$ for $1 < m \leq k$
- $(q > r_m) \wedge (m = k)$

If one of the first two cases is satisfied, the search continues on the child node found by following p_m, whereas if the last condition is satisfied, the pointer p_{m+1} is followed. Since the keys in an internal node are sorted, a binary search can be performed to find r_m. Finally, when a leaf node is reached, the entire bucket is read into memory first, then a record with a key value of q is searched.

8.2.5. Constructing a B^+-tree

The construction of a B^+-tree is performed through successive record insertions.[3] Given a record, we first compute its key, find the leaf node it is supposed to be in, and insert it if the bucket is not full. Otherwise, the leaf node is split into two nodes, each containing $\lceil bkfr/2 \rceil$, and $\lfloor bkfr/2 \rfloor + 1$ records, and the first key of the node containing the larger key values is placed into the parent internal node together with the node's pointer. The insertion of a key to an internal node is similar, only this time both split nodes contain v values, and the middle key value is sent up to the parent node.

Note that not all the internal nodes of a B^+-tree have to be kept on the disk, and read from there each time we do a search. In practice, all but the last two levels of a B^+-tree are placed in memory. The reason for this is the high branching factor of the B^+-trees together with their effective storage utilization. It has been shown in [103] that the nodes of a high-order B^+-tree are $ln2 \approx 69\%$ full on average.

However, note that the tree will be fixed in our case, i.e., once it is constructed we will not be inserting any other N-gram records. Therefore we do not need to worry about the 69% space utilization, but instead try to make each bucket, and each internal node full. Thus, with a $bkfr = 1250$, and $v = 100$, an N-gram file with 10,000,000 records would have 8,000 leaf nodes on level 3, 40 internal nodes on level 2, and the root node on level 1. Furthermore, let us assume that integers, disk and memory pointers all hold 8 bytes of space. Therefore a 5-gram key would require 40 bytes, and a full internal node in level 2 would require $(200x40) + (201x8) = 9,608$ bytes. Thus the level 2 would require $9,608x40 \approx 384$ Kbytes, and level 1 would require $(40 * 40) + (41 * 8) = 1,928$ bytes. Hence, a Web1T 5-gram file, which has an average size of 286 MB can be indexed with approximately 386 Kbytes. There are 118 5-gram files in the Web1T dataset, so we would need $386x118 \approx 46$ MBytes of memory space in order to index all of them. A similar calculation for 4-grams, trigrams, and bigrams for which the bucket factor values are selected as 1600, 2000, and 2500 respectively, shows that the entire Web1T corpus, except unigrams, can be indexed

[3]Note that this may cause efficiency issues for very large files as memory might become full during the construction process, hence in practice, the file is usually sorted prior to indexing.

with approximately 100 MBytes, all of which can be kept in memory, thereby reducing the disk access to only one. As a final note, in order to compute a key for a given N-gram quickly, we keep the unigrams in memory, and use a hashing scheme for mapping tokens to integers, which additionally require 178 Mbytes of memory space.

The choice of the bucket factor and the internal node order parameters depend on the hard-disk speed, and the available memory.[4]. Recall that even to fetch a single N-gram record from the disk, the entire bucket needs to be read. Therefore as the bucket factor parameter is reduced, the size of the index will grow, but the access time would be faster as long as the index could be entirely fit in memory. On the other hand, with a too large bucket factor, although the index can be made smaller, thereby reducing the memory requirements, the access time may be unacceptable for the application. Note that a random reading of a bucket of records from the hard-disk requires the disk head to first go to the location of the first record, and then do a sequential read.[5] Assuming a hard-disk having an average transfer rate of 100 MBytes, once the disk head finds the correct location, a 40 bytes N-gram record can be read in $4x10^{-7}$ seconds. Thus, assuming a seek time around 8-10 ms, even with a bucket factor of 1,000, it can be seen that the seek time is still the dominating factor. Therefore, as the bucket size gets smaller than 1,000, even though the index size will grow, there would be almost no speed up in the access time, which justifies our parameter choices.

8.2.6. Handling Wild Card Queries

Having described the indexing scheme, and how to search for a single N-gram record, we now turn our attention to queries including one or more wild card symbols, which in our case is the underscore character "_", as it does not exist among the unigram tokens of the Web1T dataset. We manually add the wild card symbol to our mapping of tokens to integers, and map it to the integer 0, so that a search for a query with a wild card symbol would be unsuccessful but would point to the first record in the file that replaces the wild card symbol with a real token as the key for the wild card symbol is guaranteed to be the

[4]We used a 7200 RPM disk-drive with an average read seek time of 8.5 ms, write seek time of 10.0 ms, and a data transfer time up to 3 GBytes per second.

[5]A rotational latency should also be taken into account before the sequential reading can be done.

smallest. Having found the first record we perform a sequential read until the last read record does not match the query. The reason this strategy works is because the N-grams are sorted in lexicographic order in the data set, and also when we map unigram tokens to integers, we preserve their order, i.e., the first token in the lexicographically sorted unigram list is assigned the value 1, the second is assigned 2, and so forth. For example, for a given query "Our Honorable _", the record that would be pointed at the end of search in the trigram file 3gm-0041 is the N-gram "Our Honorable Court 186", which is the first N-gram in the data set that starts with the bigram "Our Honorable".

Note however that the methodology that is described to handle the queries with wild card symbols will only work if the wild card symbols are the last tokens of the query and they are contiguous. For example a query such as "Our _ Court" will not work as N-grams satisfying this query are not stored contiguously in the data set. Therefore in order to handle such queries, we need to store additional copies of the N-grams sorted in different orders. When the last occurrence of the contiguous wild card symbols is in position p of a query N-gram for $p = 0, 1, ..., N-1$, then the N-grams sorted lexicographically starting from position $(p+1) mod N$ needs to be searched. A lexicographical sort for a position p, for $0 \leq p \leq (N-1)$ is performed by moving all the tokens in positions $0...(p-1)$ to the end for each N-gram in the data set. Thus, for all the bigrams in the data set, we need one extra copy sorted in position 1, for all the trigrams, we need two extra copies; one sorted in position 1, and another sorted in position 2, and so forth. Hence, in order to handle the contiguous wild card queries in any position, in addition to the 88 GBytes of original Web1T data, we need an extra disk space of 265 GBytes. Furthermore, the indexing cost of the duplicate data is an additional 320 MBytes. Thus, the total disk cost of the system will be approximately 353 GBytes plus the index size of 420 MBytes, and since we keep the entire index in memory, the final memory cost of the system will be 420 MBytes + 178 MBytes = 598 MBytes.

8.2.7. Performance

Given that today's commodity hardware comes with at least 4 GBytes of memory and 1 TBytes of hard-disk space, the requirements of our tool are reasonable. Furthermore, our tool is implemented in a client-server architecture, and it allows multiple clients to submit multiple queries to the server over a network. The server can be queried with an N-gram query either for its count in the corpus, or its smoothed probability with a given smoothing method. The queries with wild cards can ask for the retrieval of all the N-grams satisfying a query, or only for the total count so the network overhead can be avoided depending on the application needs.

Our program requires about one day of offline processing due to resorting the entire data a few times. Note that some of the files in the corpus need to be sorted as many as four times. For the sorting process, the files are first individually sorted, and then a k-way merge is performed. In our implementation, we used a min heap structure for this purpose, and k is always chosen as the number of files for a given N. The index construction however is relatively fast. It takes about an hour to construct the index for the 5-grams. Once the offline processing is done, it only takes a few minutes to start the server, and from that point the online performance of our tool is very fast. It takes about 1-2 seconds to process 1000 randomly picked 5-gram queries (with no wild card symbols), which may or may not exist in the corpus. For the queries asking for the frequencies only, our tool implements a small caching mechanism that takes the temporal locality into account. The mechanism is very useful for wild card queries involving stop words, such as "the _", and "of the _" which occur frequently, and take a long time to process due to the sequential read of a large number of records from the data set.

8.3. Sentence Compression Algorithm

The sentence compression algorithm presented in this chapter simply tries to get rid of the adverbial phrases in the sentence. The adverbial phrases such as "as well", or "so far" often times do not bring extra information into the summary, and hence can be discarded.

Feature	ROUGE-1	ROUGE-2	ROUGE-SU4
Linear Regression	0.484	0.086	0.173
Multi-layer Perceptron	0.488	0.089	0.178
SMOReg	0.495	0.098	0.187

TABLE 8.1. The ROUGE Recall results of the machined learned summarizers with sentence compressors on novels domain.

The adverbial phrases in a sentence are detected by using The Stanford Parser[6] [63]. Once an adverbial phrase is extracted, its n-gram probability is calculated from the Web1T Corpus using the fast access tool introduced in the previous section. The n-gram probability is calculated with absolute discounting. Once the probability p_a is known, its self information can be calculated via the following formula:

$$I(p_a) = \frac{1}{log(p_a)} = -log(p_a) \tag{19}$$

From a development set of 30 sentences, a threshold value is selected to decide on the worthiness of the adverbial phrase. Given a new adverbial phrase, if its self-information falls below this threshold, the phrase is discarded from the sentence. Otherwise it is kept as is.

8.4. Results

Compressing the sentences by removing certain adverbial phrases leads to a 17% compression ratio. Hence more room is created for other sentences in a summary. In fact this extra room leads to improvement on the supervised extraction systems of the novels domain. This improvement is summarized in Table 8.1.

8.5. Related Work

Language modeling toolkits are used extensively for speech processing, machine translation, and many other NLP applications. The two of the most popular toolkits that are also freely available are the CMU Statistical Language Modeling (SLM) Toolkit [16], and

[6]The parser is downloaded from http://nlp.stanford.edu/software/lex-parser.shtml

the SRI Language Modeling Toolkit [94]. However, even though these tools represent a great resource for building language models and applying them to various problems, they are not designed for very large corpora, such as the Web1T 5-gram corpus [9], hence they do not provide efficient implementations to access these data sets.

Furthermore, [107] has recently shown that the widely popular smoothing algorithms for language models such as Kneser-Ney [42], Witten-Bell [101], or Absolute Discounting do not realize the full potentials of very large corpora, which often come with missing counts. The reason for the missing counts is due to the omission of low frequency N-grams in the corpus. [107] shows that with a modified version of Kneser-Ney smoothing algorithm, named as the Dirichlet-Kneser-Ney, a 31% reduction in perplexity can be obtained on the Brown corpus.

A tool similar to ours that uses a hashing technique in order to provide a fast access to the Web1T corpus is presented in detail in [32]. The tool provides access to queries with wild card symbols, and the performance of the tool on 10^6 queries on a 2.66 GHz processor with 1.5 GBytes of memory is given approximately as one hour. Another tool, Web1T5-Easy, described in [25], provides indexing of the Web1T corpus via relational database tables implemented in an SQLite engine. It allows interactive searches on the corpus as well as collocation discovery. The indexing time of this tool is reported to be two weeks, while the non-cached retrieval time is given to be in order of a few seconds. Other tools that implement a binary search algorithm as a simpler, yet less efficient method are also given in [29, 106].

Sentence extraction is a well studied topic in Natural Language Processing with mostly supervised systems dominating the field [43, 98, 64, 14].

8.6. Summary

In this chapter a new publicly available tool that provides fast access to large N-gram datasets with modest hardware requirements is described. In addition to providing access to individual N-gram records, the tool also handles queries with wild card symbols, provided that the wild cards in the query are contiguous. Furthermore, the tool also implements smoothing algorithms that try to overcome the missing counts that are typical to

N-gram corpora due to the omission of low frequencies. The tool is then used in a simple sentence compression algorithm that simply calculates the self information contained in the adverbial phrases of the generated summary sentences. The adverbial phrases are either kept in or discarded from a summary sentence based on the threshold value obtained from a development set. Removing the adverbial phrases with low self information scores from the extracted summary sentences allows for more sentence extraction hence repeating this process iteratively generates more compact and higher compressed summaries. As shown in the evaluation results, this process increases the overall metrics for the system.

CHAPTER 9

CONCLUSION

This thesis addressed the problem of summarization of novels, which are long and complex literary narratives. Rather than providing a complete solution to this newly addressed problem, the main objective throughout the thesis was to break the ground on this new domain by performing a deep investigation by collecting a new data set, analyzing the problem, and finally developing models that make attempts towards a solution. While some of the features included in the models developed were borrowed from the domain of short document summarization and modified to adapt them to longer documents, some of the features were newly introduced.

Since summarization of very long documents have never been investigated before, introducing a new data set was mandatory to start tackling this problem. Hence Chapter 3 started by describing how various online resources were leveraged for this purpose. A total of 66 novels were collected, where each novel had 108,800 words, and 32 chapters on average. The summaries for the novels were provided per chapter, so each chapter of a novel, with an average length of 3,400 words, was considered in isolation. It should be noted that the average length of a document, i.e. a single chapter of a novel, is still much larger than news, medical or any other domain studied for text summarization. Furthermore, the compression ratio for the provided summaries were also higher in comparison to other domains, ranging from 87% to 96% depending on the source of the summary. Finally, Chapter 3 also introduced the ROUGE evaluation toolkit used throughout the thesis.

The summaries collected as described in 3 came in three different types; synopsis, objective, and interpretative. Synopsis summaries do not provide a summary per chapter but rather summarize the entire novel in one or two pages thereby having a very high compression ratio. Evaluation of these types of highly compressed summaries via ROUGE

or any other automatic evaluation methods developed to date cannot be relied on as the original sentences of the novel go through a great amount of transformations. Therefore decoding the information contained in a summary sentence and linking back to its original source becomes a very challenging task. Thus the synopsis type of summaries were discarded for this study.

Both the objective, and the interpretative types of summaries are in chapter level, and have lower compression ratios as described above. The difference between these two types of summaries come from the fact that the objective summaries describe the events in a novel without adding any interpratation or exploratory information from the writer of the summary. Hence in an interpretative summary, one would expect to see more transformations of the novel sentences combined with new sentences and vocabulary usage as there would be new information added by the summary writer. This intuition was analyzed in Chapter 4 by using a sentence decomposition algorithm.

The goal of the sentence decomposition algorithm used in Chapter 4 was to link a summary sentence back to its source sentences in the novel by identifying the transformational operations, such as cut-and-paste. Knowing the percentage of summary sentences that can be linked, it can be inferred whether a summary type is suitable for extractive summarization or not. The result of this analysis in Chapter 4 showed that the percentage of the objective summary sentences linked back to the their sources by the decomposition algorithm was about twice the size of the percentage found for the interpretative summaries. The conclusion was as expected; that humans use very little extraction from the source document when writing interpretative summaries, and thus extractive summarization is not suitable for this summary type. On the other hand, it was found that on average, around 30% of the human-written objective summary sentences are constructed through cut-and-paste operations from the source document, which made this type of summaries worthy of further investigation.

Having selected the objective summaries as ground truth for evaluation, Chapter 5 used them to perform a different type of analysis in which the objective was to understand

how successful one can be by only doing extractive summarization in this new domain, and how that compares with other domains. In order to reach this goal, the search space of extractive summaries were explored across four different domains. This was achieved by generating every possible summary for a document and then evaluating each of these summaries via ROUGE againist the ground truth summary. Then the distributions of the evaluation scores were used to generate the probability density functions (pdfs). The pdf for a document was used to calculate the percentile rank of a given summary, which indicate how well a given summary performs out of all the possible summaries for that document. By combining pdfs obtained for a corpus of documents selected from a domain, it is also possible to evaluate the performance of a summarization system for that domain in the same way. Furthermore, when the entire domains are considered, the percentile rank indicates how difficult it would be to move the needle for a given system which cannot be inferred from the absolute score distance from the upper bound. For example, for the news domain while there is a 20.1% gap between the upper bound and the lead baseline, the percentile rank of 99.99% for the lead baseline suggests that closing this gap is difficult as the system already performs significantly above the average.

The analysis in Chapter 5 for the literary novels revealed that extractive summarization in this domain is a harder task than both the news, and scientific domains but easier than the legal domain as indicated by the standard deviations of the corresponding pdfs. Furthermore, looking at the three extractive summarization systems used on these domains; it can be seen that the lead baseline, which is very successful in the news domain, behaves just like an algorithm that randomly selects sentences for the novels domain, meaning that the lead sentences of a novel chapter have no importance, hence are not good candidates for novel summaries. However, using the TextRank graph centrality algorithm, a much smarter way of selecting the summary sentences, proved quite successful by bringing the percentile rank for ROUGE-1 to 97.9%.

After concluding in Chapter 5 that there is room for improvement in extractive summarization of novels, and smarter algorithms lead to better results, Chapter 6 made the

first attempt towards building extractive summarization systems in an unsupervised fashion. When the features used in these systems take into account the characteristics of long documents, the systems achieved superior performance compared to those that give state of the art results in news documents which are designed for shorter documents of a completely different domain. Thus the final system developed in this chapter achieved a significant improvement over MEAD by combining the following features: The positional scores were discarded, the sentences are ranked using TextRank and were assigned a score based on their importance, and the sentences were also segmented using a segmentation algorithm, and the segment, and inverse segment frequency of a word was taken into account together with its frequency in the chapter and its inverse document frequency obtained from an external corpus. Moreover the segments of a chapter were ranked using TextRank, which assigned each segment a score based on its importance, and the sentences in each segment were weighted with this score.

In Chapter 6, the features developed were combined in an ad-hoc way so the resulting system was unsupervised. Hence the next step was to experiment with supervised methods in order to come up with better systems, which was the goal in Chapter 7. The training data for the supervised methods came from the analysis performed in Chapter 5. The supervised framework developed in this chapter treated each individual feature as a sentence ranker for a given document, hence the objective function the system tried to optimize by making use of the training data was a sentence ranking function that combined various individual features including the position of the sentence within the document, the length of the sentence, the term frequency score, the TextRank score, the number of named entities, etc. All the individual features as well as the new system was evaluated on two different domains; news and literary novels, which provided another view on the importance of each feature depending on the domain. The results showed that most of the supervised systems built using the given framework gave better results than each of the individual unsupervised features.

It should however be noted that the training data used in this framework was not ideal in the sense that the labeling of a training instance was based on the average ROUGE evaluation score of all the summaries that contain the sentence, rather than the evaluation score of the sentence itself. Hence the individual contribution of the sentence to the overall evaluation score was not known. With this approximation, the score of a high quality sentence that should be included in a summary would still be higher than a low quality sentence but the gap would be lowered due to the effect of averaging. Hence an evaluation metric that scores the sentences on an individual basis is expected to make the supervised framework described in Chapter 7 even more successful.

The final chapter, Chapter 8, broke away from the extractive summarization models, and considered sentence compression, which is typically investigated in the interpretation stage of automatic text summarization. The goal of this chapter was to see the effect of sentence compression to the overall evaluation metrics. Since the summaries for the literary novels are much longer in length than other domains such as news, an iterative process that first compresses the sentences in a summary and then adds new ones as long as the compression ratio allows is ultimately expected to increase the overall metrics, as the summaries become more compact and contain more information. This was shown with a new yet fairly simple compression algorithm. The algorithm used a probabilistic parser in order to identify the adverbial phrases in a sentence, and then used a newly developed tool, which provides fast access to large N-gram datasets, in order to compute the self information contained in the adverbial phrases. An adverbial phrase is discarded if the algorithm decides that it has very little self information. This process led to an overall 17% compression on the generated sentences and caused the overall metrics to increase significantly.

In conclusion, the thesis provided a detailed study on many different aspects of a brand new problem, the extractive summarization of literary novels, while also briefly touching the interpretation stage in Chapter 8. The very last stage of automatic text summarization, i.e. generation, which attempts to bring the summaries into a more coherent and human-readable form is not addressed in this thesis as the task requires a detailed study of its own.

Hence a more detailed investigation of the interpretation, and generation stages are left as a future study.

BIBLIOGRAPHY

[1] C. Aone, J. Gorlinsky, B. Larsen, and M. E. Okurowski, *A trainable summarizer with knowledge acquired from robust nlp techniques*, In I. Mani and M. Maybury (eds), Advances in Automated Text Summarization, MIT Press, 1999, p. 3.

[2] Michele Banko and Lucy Vanderwende, *Using n-grams to understand the nature of summaries*, Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (Boston, Massachusetts, USA), 2004.

[3] Regina Barzilay and Michael Elhadad, *Using lexical chains for text summarization*, In Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization, 1997, pp. 10–17.

[4] Regina Barzilay and Lillian Lee, *Catching the drift: Probabilistic content models, with applications to generation and summarization*, HLT-NAACL 2004: Proceedings of the Main Conference, 2004, pp. 113–120.

[5] Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad, *Information fusion in the context of multi-document summarization*, Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics (Morristown, NJ, USA), Association for Computational Linguistics, 1999, pp. 550–557.

[6] P. B. Baxendale, *Machine-made index for technical literature – an experiment*, IBM Journal of Reasearch and Development 3 (1958), no. 2, 354361.

[7] Steven Bird, Ewan Klein, and Edward Loper, *Natural Language Processing with Python*, O'Reilly Media, 2009.

[8] Harold Borko and Charles Bernier, *Abstracting concepts and methods*, Academic Press, New York, 1975.

[9] T. Brants and A. Franz, *Web 1T 5-gram corpus version 1*, Linguistic Data Consortium (2006).

[10] Hakan Ceylan and Rada Mihalcea, *The decomposition of human-written book summaries*, CICLing '09: Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing (Berlin, Heidelberg), Springer-Verlag, 2009, pp. 582–593.

[11] ———, *An efficient indexer for large n-gram corpora*, Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations (Stroudsburg, PA, USA), HLT '11, Association for Computational Linguistics, 2011, pp. 103–108.

[12] Hakan Ceylan, Rada Mihalcea, Umut Özertem, Elena Lloret, and Manuel Palomar, *Quantifying the limits and success of extractive summarization systems across domains*, Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Stroudsburg, PA, USA), HLT '10, Association for Computational Linguistics, 2010, pp. 903–911.

[13] Freddy Y. Y. Choi, *Advances in domain independent linear text segmentation*, In Proceedings of NAACL, 2000, pp. 26–33.

[14] James Clarke and Mirella Lapata, *Models for sentence compression: a comparison across domains, training requirements and evaluation measures*, ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (Morristown, NJ, USA), Association for Computational Linguistics, 2006, pp. 377–384.

[15] James Clarke and Mirella Lapata, *Modelling compression with discourse constraints*, Proceedings of the Conference on Empirical Methods in Natural Language Processing and on Computational Natural Language Learning (EMNLP-CoNLL-2007) (Prague, Czech Republic), 2007, pp. 1–11.

[16] Philip Clarkson and Ronald Rosenfeld, *Statistical language modeling using the cmu-cambridge toolkit*, Proceedings of ESCA Eurospeech, 1997, pp. 2707–2710.

[17] Edward T. Cremmins, *The art of abstracting*, 2nd ed., Arlington, Va. : Information Resources Press, 1996.

[18] Hal Daumé III and Daniel Marcu, *A phrase-based HMM approach to document/abstract alignment*, Proceedings of EMNLP (Barcelona, Spain), 2004.

[19] ———, *Induction of word and phrase alignments for automatic document summarization*, Computational Linguistics 31 (2005), no. 4, 505–530.

[20] E. D'Avanzo and B. Magnini, *A keyphrase-based approach to summarization: The Lake system at DUC 2005*, Proceedings of the Document Understanding Conference (DUC 2005), 2005.

[21] Robert L. Donaway, Kevin W. Drummey, and Laura A. Mather, *A comparison of rankings produced by summarization evaluation measures*, NAACL-ANLP 2000 Workshop on Automatic summarization (Morristown, NJ, USA), Association for Computational Linguistics, 2000, pp. 69–78.

[22] H. P. Edmundson, *New methods in automatic extracting*, J. ACM 16 (1969), no. 2, 264–285.

[23] Brigitte Endres-niggemeyer and Elisabeth Neugebauer, *Professional summarising: No cognitive simulation without observation*, In Proceedings of the International Conference in Cognitive Science, 1995.

[24] G. Erkan and Dragomir R. Radev, *Lexrank: Graph-based centrality as salience in text summarization*, Journal of Artificial Intelligence Research 22 (2004).

[25] Stefan Evert, *Google web 1t 5-grams made easy (but not for the computer)*, Proceedings of the NAACL HLT 2010 Sixth Web as Corpus Workshop, WAC-6 '10, 2010, pp. 32–40.

[26] Christiane Fellbaum, *Wordnet: An electronic lexical database*, Bradford Books, 1998.

[27] Udo Fries, *Summaries in newspapers: A textlinguistic investigation*, The Structure of Texts (Udo Fries, ed.), Gunter Narr Verlag Tbingen, 1997, pp. 47–63.

[28] M. Galley, *Automatic summarization of conversational multi-party speech*, Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006), AAAI/SIGART Doctoral Consortium (Boston), 2006.

[29] Claudio Giuliano, Alfio Gliozzo, and Carlo Strapparava, *Fbk-irst: lexical substitution task exploiting domain and syntagmatic coherence*, SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations, 2007, pp. 145–148.

[30] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten, *The weka data mining software: an update*, SIGKDD Explor. Newsl. 11 (2009), 10–18.

[31] Michael A. K. Halliday and Ruqaiya Hasan, *Cohesion in English*, English Language Series, Longman Group Ltd, 1976.

[32] Tobias Hawker, Mary Gardiner, and Andrew Bennetts, *Practical queries of a massive n-gram database*, Proceedings of the Australasian Language Technology Workshop 2007 (Melbourne, Australia), December 2007, pp. 40–48.

[33] Marti A. Hearst, *Multi-paragraph segmentation of expository text*, Proceedings of the 32nd annual meeting on Association for Computational Linguistics (Morristown, NJ, USA), Association for Computational Linguistics, 1994, pp. 9–16.

[34] T. Hirao, Y. Sasaki, H. Isozaki, and E. Maeda, *NTT's text summarization system for DUC-2002*, Proceedings of the Document Understanding Conference 2002 (DUC 2002), 2002.

[35] Eduard Hovy and Chin-Yew Lin, *Automated text summarization and the summarist system*, Proceedings of a workshop on held at Baltimore, Maryland (Morristown, NJ, USA), Association for Computational Linguistics, 1996, pp. 197–214.

[36] Eduard Hovy, Chin yew Lin, and Liang Zhou, *Evaluating duc 2005 using basic elements*, Proceedings of DUC-2005, 2005.

[37] Eduard H. Hovy and Chin Yew Lin, *Automated text summarization in summarist*, Advances in Automatic Text Summarization (Inderjeet Mani and Mark T. Maybury, eds.), MIT Press, 1999, pp. 81–97.

[38] Hongyan Jing, *Using hidden markov modeling to decompose human-written summaries*, Comput. Linguist. 28 (2002), no. 4, 527–543.

[39] Hongyan Jing and Kathleen R. McKeown, *The decomposition of human-written summary sentences*, SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 1999, pp. 129–136.

[40] A. Kazantseva, *An approach to summarizing short stories.*, Proceedings of the Student Workshop at the European Association for Computational Linguistics (Trento, Italy), 2006.

[41] A. Kazantseva and S. Szpakowicz, *Challenges in evaluating summaries of short stories*, Proceedings of the Workshop on Task-Focused Summarization and Question Answering (Sydney, Australia), 2006, pp. 8–15.

[42] R. Kneser and H. Ney, *Improved backing-off for n-gram language modeling*, Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on, vol. 1, 1995, pp. 181–184 vol.1.

[43] Kevin Knight and Daniel Marcu, *Statistics-based summarization - step one: Sentence compression*, Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, AAAI Press / The MIT Press, 2000, pp. 703–710.

[44] Julian Kupiec, Jan Pedersen, and Francine Chen, *A trainable document summarizer*, SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 1995, pp. 68–73.

[45] Frederick W. Lancaster, *Indexing and abstracting in theory and practice*, University of Illionis Graduate School of Library and Information Science, Champaign, Illionis, 1991.

[46] Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu, *Enhancing diversity, coverage and balance for summarization through structure learning*, Proceedings of the 18th international conference on World wide web (New York, NY, USA), WWW '09, ACM, 2009, pp. 71–80.

[47] W. Li, W. Li, B. Li, Q. Chen, and M. Wu, *The Hong Kong Polytechnic University at DUC 2005*, Proceedings of the Document Understanding Conference (DUC 2005) (Vancouver, Canada), 2005.

[48] Chin Y. Lin, *Topic identification by concept generalization*, Proceedings of 33rd ACL Conference (Boston, MA), June 1995.

[49] Chin-Yew Lin, *Training a selection function for extraction*, CIKM '99: Proceedings of the eighth international conference on Information and knowledge management (New York, NY, USA), ACM, 1999, pp. 55–62.

[50] ———, *Rouge: A package for automatic evaluation of summaries*, Text Summarization Branches Out: Proceedings of the ACL-04 Workshop (Barcelona, Spain) (Stan Szpakowicz Marie-Francine Moens, ed.), Association for Computational Linguistics, July 2004, pp. 74–81.

[51] Chin-Yew Lin and Eduard Hovy, *Identifying topics by position*, Proceedings of the fifth conference on Applied natural language processing (Morristown, NJ, USA), Association for Computational Linguistics, 1997, pp. 283–290.

[52] ———, *The potential and limitations of automatic sentence extraction for summarization*, Proceedings of the HLT-NAACL 03 on Text summarization workshop (Morristown, NJ, USA), Association for Computational Linguistics, 2003, pp. 73–80.

[53] Elena Lloret and Manuel Palomar, *A gradual combination of features for building automatic summarisation systems*, TSD '09: Proceedings of the 12th International Conference on Text, Speech and Dialogue (Berlin, Heidelberg), Springer-Verlag, 2009, pp. 16–23.

[54] ———, *Challenging issues of automatic summarization: Relevance detection and quality-based evaluation*, Informatica (Slovenia) 34 (2010), no. 1, 29–35.

[55] H. P. Luhn, *The automatic creation of literature abstracts*, IBM J. Res. Dev. 2 (1958), 159–165.

[56] I. Malioutov and R. Barzilay, *Minimum cut model for spoken lecture segmentation*, Proceedings of the Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006), 2006, pp. 9–16.

[57] Inderjeet Mani, *Automatic summarization*, Natural Language Processing, John Benjamins Publishing Company, 2001.

[58] Inderjeet Mani, Barbara Gates, and Eric Bloedorn, *Improving summaries by revising them*, Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics (Morristown, NJ, USA), Association for Computational Linguistics, 1999, pp. 558–565.

[59] Inderjeet Mani and Mark T. Maybury, *Advances in automatic text summarization*, MIT Press, 1999.

[60] D. Marcu, *Improving summarization through rhetorical parsing tuning*, Proceedings od the COLING-ACL: The Sixth Workshop on Very Large Corpora, 1998, pp. 10–16.

[61] Daniel Marcu, *The automatic construction of large-scale corpora for summarization research*, SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 1999, pp. 137–144.

[62] ———, *The theory and practice of discourse parsing and summarization*, MIT Press, Cambridge, MA, USA, 2000.

[63] M. Marneffe, B. Maccartney, and C. Manning, *Generating Typed Dependency Parses from Phrase Structure Parses*, Proceedings of LREC-06, 2006, pp. 449–454.

[64] Ryan McDonald, *Discriminative Sentence Compression with Soft Syntactic Constraints*, Proceedings of the 11th˜EACL (Trento, Italy), 2006.

[65] Kathleen McKeown and Dragomir R. Radev, *Generating summaries of multiple news articles*, SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 1995, pp. 74–82.

[66] Rada Mihalcea, *Language independent extractive summarization*, Proceedings of the ACL Interactive Poster and Demonstration Sessions (Ann Arbor, Michigan), Association for Computational Linguistics, June 2005, pp. 49–52.

[67] Rada Mihalcea and Hakan Ceylan, *Explorations in automatic book summarization*, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL) (Prague, Czech Republic), Association for Computational Linguistics, June 2007, pp. 380–389.

[68] Rada Mihalcea and Paul Tarau, *Textrank: Bringing order into texts*, Conference on Empirical Methods in Natural Language Processing (Barcelona, Spain), 2004.

[69] Maria Pinto Molina, *Documentary abstracting: toward a methodological model*, J. Am. Soc. Inf. Sci. 46 (1995), no. 3, 225–234.

[70] Tatsunori Mori, *Information gain ratio as term weight: The case of summarization of ir results*, Proceedings of the 19th International Conference on Computational Linguistics (COLING 02), 2002, pp. 688–694.

[71] Jane Morris and Graeme Hirst, *Lexical cohesion computed by thesaural relations as an indicator of the structure of text*, Comput. Linguist. 17 (1991), 21–48.

[72] Yoshio Nakao, *An algorithm for one-page summarization of a long text based on thematic hierarchy detection*, ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (Morristown, NJ, USA), Association for Computational Linguistics, 2000, pp. 302–309.

[73] Ani Nenkova, *Automatic text summarization of newswire: lessons learned from the document understanding conference*, AAAI'05: Proceedings of the 20th national conference on Artificial intelligence, AAAI Press, 2005, pp. 1436–1441.

[74] Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown, *The pyramid method: Incorporating human content selection variation in summarization evaluation*, ACM Trans. Speech Lang. Process. 4 (2007), no. 2, 4.

[75] Miles Osborne, *Using maximum entropy for sentence extraction*, Proceedings of the ACL-02 Workshop on Automatic Summarization (Morristown, NJ, USA), Association for Computational Linguistics, 2002, pp. 1–8.

[76] P. Over and W. Liggett, *Introduction to duc: An intrinsic evaluation of generic news text summarization systems*, Proceedings of the DUC 2002 Workshop on Text Summarization, 2002.

[77] Karolina Owczarzak, *Depeval(summ): dependency-based evaluation for automatic summaries*, ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 (Morristown, NJ, USA), Association for Computational Linguistics, 2009, pp. 190–198.

[78] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd, *The pagerank citation ranking: Bringing order to the web*, Tech. report, Stanford InfoLab, 1999.

[79] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, *Bleu: a method for automatic evaluation of machine translation*, Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (Stroudsburg, PA, USA), ACL '02, Association for Computational Linguistics, 2002, pp. 311–318.

[80] D. Radev, J. Otterbacher, H. Qi, and D. Tam, *MEAD ReDUCs: Michigan at DUC 2003*, Proceedings of the Document Understanding Conference (DUC 2003), 2003.

[81] Dragomir R. Radev, Eduard Hovy, and Kathleen McKeown, *Introduction to the special issue on summarization*, Comput. Linguist. 28 (2002), no. 4, 399–408.

[82] Dragomir R. Radev, Hongyan Jing, Malgorzata Styś, and Daniel Tam, *Centroid-based summarization of multiple documents*, Inf. Process. Manage. 40 (2004), no. 6, 919–938.

[83] Dragomir R. Radev and Kathleen R. McKeown, *Generating natural language summaries from multiple on-line sources*, Comput. Linguist. 24 (1998), no. 3, 470–500.

[84] Ulrich Reimer and Udo Hahn, *A formal model of text summarization based on condensation operators of a terminological logic*, In I. Mani and M. Maybury (eds), Advances in Automated Text Summarization, MIT Press, 1997, p. 3.

[85] Horacio Saggion, Simone Teufel, Dragomir Radev, and Wai Lam, *Meta-evaluation of summaries in a cross-lingual environment using content-based metrics*, Proceedings of the 19th international conference on Computational linguistics (Morristown, NJ, USA), Association for Computational Linguistics, 2002, pp. 1–7.

[86] Gerard Salton and Christopher Buckley, *Term-weighting approaches in automatic text retrieval*, Inf. Process. Manage. 24 (1988), 513–523.

[87] Gerard Salton, Amit Singhal, Mandar Mitra, and Chris Buckley, *Automatic text structuring and summarization*, Inf. Process. Manage. 33 (1997), no. 2, 193–207.

[88] Betty Salzberg, *File structures: an analytic approach*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[89] Frank Schilder and Ravikumar Kondadadi, *Fastsum: fast and accurate query-based multi-document summarization*, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers (Stroudsburg, PA, USA), HLT-Short '08, Association for Computational Linguistics, 2008, pp. 205–208.

[90] Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen, *Document summarization using conditional random fields*, Proceedings of the 20th international joint conference on Artifical intelligence (San Francisco, CA, USA), Morgan Kaufmann Publishers Inc., 2007, pp. 2862–2867.

[91] Alex J. Smola and Bernhard Schölkopf, *A tutorial on support vector regression*, Statistics and Computing 14 (2004), 199–222.

[92] K. Sparck-jones, *Automatic summarising: The state of the art*, Information Processing & Management 43 (2007), no. 6, 1449–1481.

[93] Karen Sparck-Jones, *Automatic summarising: Factors and directions*, Advances in Automatic Text Summarization (Inderjeet Mani and Mark T. Maybury, eds.), MIT Press, 1999, pp. 1–13.

[94] Andreas Stolcke, *SRILM – an extensible language modeling toolkit*, Proceedings of ICSLP (Denver, USA), vol. 2, 2002, pp. 901–904.

[95] Krysta Svore, Lucy Vanderwende, and Christopher Burges, *Enhancing single-document summarization by combining RankNet and third-party sources*, Proceedings of EMNLP-CoNLL (Prague, Czech Republic), 2007, pp. 448–457.

[96] H. Tanaka, T. Kumano, M. Nishiwaki, and T. Itoh, *Analysis and modeling of manual summarization of japanese broadcast news*, Proceedings of the International Joint Conference on Natural Language Processing (Jeju Island, Korea), 2005.

[97] Simone Teufel and Marc Moens, *Sentence extraction as a classification task*, Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scallable Text Summarization (Madrid, Spain), July 1997.

[98] Jenine Turner and Eugene Charniak, *Supervised and unsupervised learning for sentence compression*, Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (Stroudsburg, PA, USA), ACL '05, Association for Computational Linguistics, 2005, pp. 290–297.

[99] Masao Utiyama and Hitoshi Isahara, *A statistical model for domain-independent text segmentation*, In Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics, 2001, pp. 491–498.

[100] S. Wan and K. McKeown, *Generating overview summaries of ongoing email thread discussions*, Proceedings of the 20th International Conference on Computational Linguistics (Geneva, Switzerland), 2004.

[101] Ian H. Witten and Timothy C. Bell, *The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression*, IEEE Transactions on Information Theory 37 (1991), no. 4, 1085–1094.

[102] F. Wolf and E. Gibson, *Paragraph-, word-, and coherence-based approaches to sentence ranking: A comparison of algorithm and human performance*, Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2004) (Barcelona, Spain), July 2004, pp. 383–390.

[103] Andrew Chi-Chih Yao, *On random 2-3 trees*, Acta Inf. 9 (1978), 159–170.

[104] Min yen Kan, Judith L. Klavans, and Kathleen R. McKeown, *Linear segmentation and segment significance*, In Proceedings of the 6th International Workshop of Very Large Corpora, 1998, pp. 197–205.

[105] Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki, *Multi-document summarization by maximizing informative content-words*, Proceedings of the 20th international joint conference on Artifical intelligence (San Francisco, CA, USA), Morgan Kaufmann Publishers Inc., 2007, pp. 1776–1782.

[106] Deniz Yuret, *Ku: word sense disambiguation by substitution*, SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations, 2007, pp. 207–213.

[107] ———, *Smoothing a tera-word language model*, HLT '08: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies, 2008, pp. 141–144.

[108] L. Zhou and E. Hovy, *A Web-trained extraction summarization system*, Proceedings of Human Language Technology Conference (HLT-NAACL 2003) (Edmonton, Canada), May 2003.

[109] ———, *Digesting virtual "geek" culture: The summarization of technical internet relay chats*, Proceedings of Association for Computational Linguistics (ACL 2005) (Ann Arbor), 2005.