NTNU
Norwegian University of
Science and Technology

Extraction-Based Automatic Summarization

Theoretical and Empirical Investigation of Summarization Techniques

Gleb Sizov

Master of Science in Computer Science
Submission date: June 2010
Supervisor: Pinar Öztürk, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Summarization has always been an important task. With the swift increase in the amount of available information on the Web every day it has been even more important than the times manual summarization was feasible. Automatic summarization appears as a good candidate to resolve the information overload problem in an effective way. This project investigates approaches to generation of a summary from multiple documents/texts. It will first provide a detailed survey of summarization approaches. For this purpose, a set of important dimensions will be formulated along which various summarization approaches can be categorized, analyzed, and evaluated. The project will then focus on the design and implementation of a framework that can be used to evaluate and compare the performance of different approaches. The framework will include several of the methods for different stages/subtasks of summarization.

Assignment given: 15. January 2010
Supervisor: Pinar Öztürk, IDI

Contents

Abstract

A summary is a shortened version of a text that contains the main points of the original content. Automatic summarization is the task of generating a summary by a computer. For example, given a collection of news articles for the last week an automatic summarizer is able to create a concise overview of the important events. This summary can be used as the replacement for the original content or help to identify the events that a person is particularly interested in. Potentially, automatic summarization can save a lot of time for people that deal with a large amount of textual information. The straightforward way to generate a summary is to select several sentences from the original text and organize them in way to create a coherent text. This approach is called extraction-based summarization and is the topic of this thesis. Extraction-based summarization is a complex task that consists of several challenging subtasks. The essential part of the extraction-based approach is identification of sentences that contain important information. It can be done using graph-based representations and centrality measures that exploit similarities between sentences to identify central sentences.

This thesis provide a comprehensive overview of methods used in extraction-based automatic summarization. In addition, several general natural language processing issues such as feature selection and text representation models are discussed with regard to automatic summarization. Part of the thesis is dedicated to graph-based representations and centrality measures used in extraction-based summarization. Theoretical analysis is reinforced with the experiments using the summarization framework implemented for this thesis. The task for the experiments is query-focused multi-document extraction-based summarization, that is, summarization of several documents according to a user query. The experiments investigate several approaches to this task as well as the use of different representation models, similarity and centrality measures. The obtained results indicate that use of graph centrality measures significantly improves the quality of generated summaries. Among the variety of centrality measure the degree-based ones perform better than path-based measures. The best performance is achieved when centralities are combined with redundancy removal techniques that prevent inclusion of similar sentences in a summary. Experiments with representation models reveal that a simple local term count representation performs better than the distributed representation based on latent semantic analysis, which indicates that further investigation of distributed representations in regard to automatic summarization is necessary. The implemented system performs quite good compared with the systems that participated in DUC 2007[1] summarization competition. Nevertheless, manual inspection of the generated summaries demonstrate some of the flaws of the implemented summarization mechanism that can be addressed by introducing advanced algorithms for sentence simplification and sentence ordering.

[1]Document Understanding Conference (DUC) have been the organizer of the annual automatic summarization competition until the year 2008, starting from 2008 the competition is organized as part of Text Analysis Conference (TAC).

1 Introduction

The amount of information available today is tremendous and the problem of finding the relevant pieces and making sense of these is becoming more and more essential. Nowadays, a great deal of information comes from the Internet in a textual form. The challenge of finding relevant documents on the web is mainly handled by information retrieval techniques utilized in search engines such as Google, Bing, Yahoo, etc. Search engines usually return thousands of pages for a single query, and even the use of sophisticated ranking algorithms can't provide us the exact information we are looking for. A typical user goes through the top-ranked pages and tries to find the relevant pieces of information he or she is interested in, manually. Obviously, a short summary of the retrieved pages would be very helpful in such situations. In general, construction of summaries is an ideal way to cope with the information overload.

A summary is a shortened version of a text that contains the main points of the original content. Automatic summarization is the creation of a summary by a computer program. Although automatic summarization is a hot topic of research nowadays, only very few software tools are available to the end users and none of them are particularly popular. The reason for this should be the low quality of automatically produced summaries. In general, creation of a good summary requires a lot of intelligence. A clear evidence of this is the use of summaries in educational process. Students, especially in the humanitarian disciplines, are often required to produce clear, coherent summaries to show their understanding of the studied material. Like many other natural language processing (NLP) tasks, a high quality automatic summarization will require understanding of a natural language, at least to a certain degree. This is known to be an AI-complete task, that is, it requires a computer with human-level intelligence that strong AI claims. Despite philosophical discussions about possibilities for strong AI, even with existing methods, scientists could achieve decent results in NLP tasks that are quite challenging, such as machine translation, speech recognition, domain specific question answering, etc. Although none of these problems are near to be solved yet, the results are promising to be useful. Improving the quality of automatic summarization to this level of usefulness is the motivation behind the increasing amount of research in the field, including mine.

There are two general classes of summarization tasks: abstraction-based summarization and extraction-based summarization. In the abstraction-based summarization an abstract is created by interpreting the information contained in the original source and generating a text that expresses the same information in a more concise way, omitting insignificant details. Abstraction-based summarization is more natural for manually produced summaries. In extraction-based summarization an extract is constructed by selecting pieces of text (words, phrases, sentences, paragraphs) from the original source and organizing them in a way to produce a coherent summary. Although a high-quality abstraction-based summarizer will potentially be more useful, the research in automatic summarization is mainly focused on extraction-based methods because they employ a more straightforward approach for constructing summaries. This thesis addresses extraction-based automatic summarization.

Extraction-based summarization makes use of various techniques developed through years of research in NLP. Most of these techniques have been studied in the context of information retrieval tasks. Summarization requires additional techniques for dealing with quite specific problems such as finding the most important information within a document or a collection of documents, elimination of redundant information contained in several documents and simplification and fusion of sentences. This makes the number of techniques that can potentially be utilized for summarization to be very large. However, books or even comprehensive surveys on automatic summarization techniques are rather scarce. One of the objectives for the thesis is to provide a consistent overview of the methods used in extraction-based automatic summarization by reviewing a large number of papers in the field.

The core step of extraction-based automatic summarization is the detection of the content that should be kept in the summary, which is often referred to as *content selection* subtask. An interesting and effective way to deal with this subtask is to use graphs. Graph representations and centrality measures are often employed to identify the most influential people in a social network or informative web pages on the Web. It has been shown that similar graph-based techniques can be used for the content selection subtask as well. In the thesis some of these techniques will be investigated in detail.

Besides providing a theoretical overview of summarization techniques, we implemented our own extraction-based summarization framework (EBSF), which utilizes several different techniques for the content selection subtask including graph-based ones. This framework is used to conduct a series of experiments with the primary goal of finding out which of the applied methods contribute significantly to the quality of automatically generated summaries.

1.1 Objectives and Hypotheses

The objectives and hypotheses for the thesis are formulated as follows.

Objectives:

1. Study the known approaches to extraction-based automatic summarization and understand the difficulties and challenges on the way to a high quality summarizer.
2. Investigate the suitability of graph-based representations and algorithms for extraction-based summarization.
3. Design, implement and evaluate an extraction-based automatic summarization framework. Use the system to experiment with several different summarization methods.

Hypotheses:

1. Most of the techniques used for automatic summarization were not initially designed for this task. Even though they might have been used successfully in other NLP tasks, they may not necessarily be appropriate for the task of automatic summarization. Therefore, special adaptation of such techniques is required.
2. Graph-based representations are able to capture symbol-to-symbol relations in a more explicit way than traditional vector space models. These relations may be exploited to identify parts of a document with the most important information, which is often considered to be the core subtask in extraction-based automatic summarization.

1.2 Outline

This report comprises five chapters in addition to the introduction and conclusion. In chapter 2 a profound overview of the methods used in extraction-based automatic summarization is provided. Chapter 3 contains a discussion of graph representations and centrality measures with regard to summarization. Chapter 3 describes the extraction-based summarization framework, EBSF, implemented as part of this thesis. Chapter 4 is dedicated to the evaluation method used for the experiments in this thesis. In chapter 5 the conducted experiments are described and the experimental results are analyzed. The thesis concludes with a summary of experimental results and future research directions.

2 Survey of Summarization Methods

2.1 Classification of Summarization Tasks

The goal of automatic summarization is the construction of a concise and coherent summary of one or several documents. The task of automatic summarization can vary depending on several criteria such as the type of the produced summaries, the number of source documents, the use of external resources and the task-specific constraints. The diagram in figure 2.1 shows different kinds of summarization tasks depending on the mentioned criteria. The rest of this subsection explains each of the shown tasks.

As has been mentioned in the introduction, a summary can either be an *abstract* or an *extract*. Depending on whether we want to produce an abstract or an extract summary, the summarization process will be *abstraction-based* or *extraction-based* respectively.

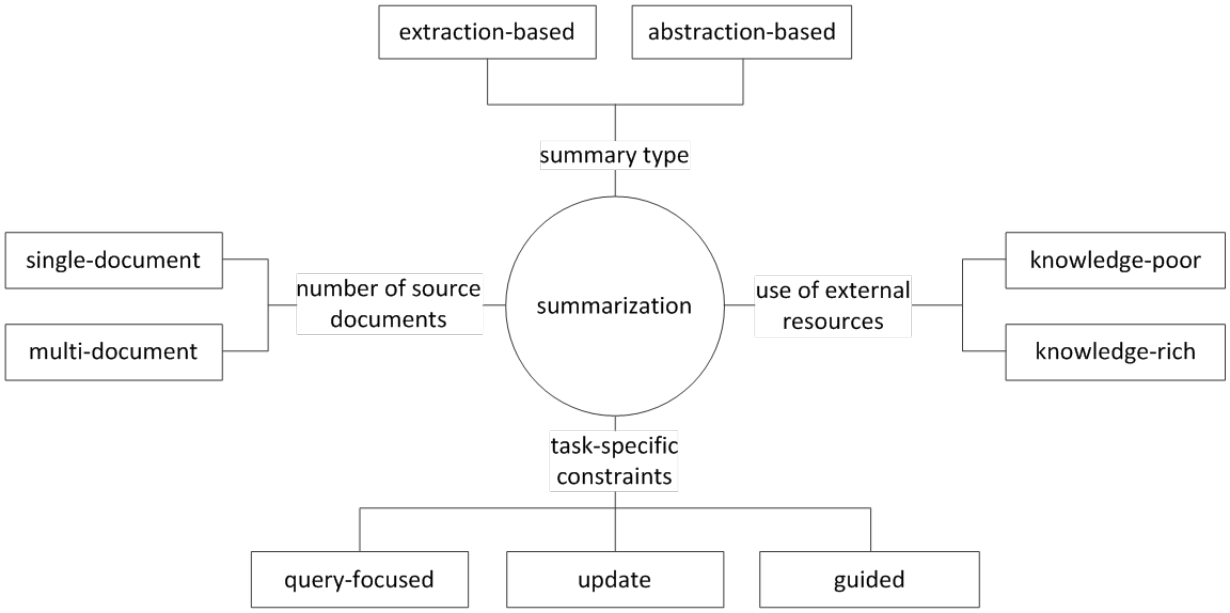

Figure 2.1: Classification of summarization tasks.

Generation of a summary of one document is often referred to as *single-document summarization* while of several documents as *multi-document summarization*. Multi-document summarization is obviously a more complex task than the single-document summarization. There are two major reasons for this. First, information overlap between the documents can lead to redundancy in the summary. Secondly, an extra effort is required to organize the information from several documents to a coherent summary.

Another criterion for classification of summarization methods is the use of external resources. *Knowledge-poor* techniques don't use any external resources while *knowledge-rich* techniques may utilize external corpus such as Wikipedia or lexical resources such as WordNet [1] or VerbOcean [2]. Such resources are often used to unravel semantic relations between words, phrases or sentences.

There are several extensions of a classical summarization task. In *query-focused* or *query-oriented summarization* a query is provided to a summarizer in addition to the source documents. The summarizer is supposed to construct a summary that contains information requested by the query. A document retrieval system together with a query-oriented multi-document summarization system is potentially a very powerful combination, which might be much more effective than a document retrieval system alone. Another extension is the so called *update summarization*. The idea of update summaries comes from the experiments with summarization of news articles that are chronologically organized. The purpose of the update summary is to identify new pieces of information in the more recent articles with the assumption that the user has already read the previous ones [3].

Very recent extension of the summarization task is the *guided summarization* [4]. In guided summarization a set of aspects that should be covered in a summary is provided. The documents are classified into several categories such as accidents, natural disasters, criminal attacks etc. Aspects vary depending on the category. For example, the summary of documents related to an accident should cover the following aspects: what happened, data, location, reason for the accident, casualties, damages and rescue efforts. The motivation for guided summarization is to inspire innovative approaches that rely on deeper linguistic analysis rather than superficial statistical features such as term frequencies.

2.2 Summarization Workflow

Automatic summarization is a multistage process. Figure 2.2 demonstrates a general extraction-based summarization workflow. Before describing each of the subtasks in the workflow it is important to clarify that the term *context*, which appears in the workflow and further in this report quite frequently, refers to a piece of text smaller than a document but larger than a single word

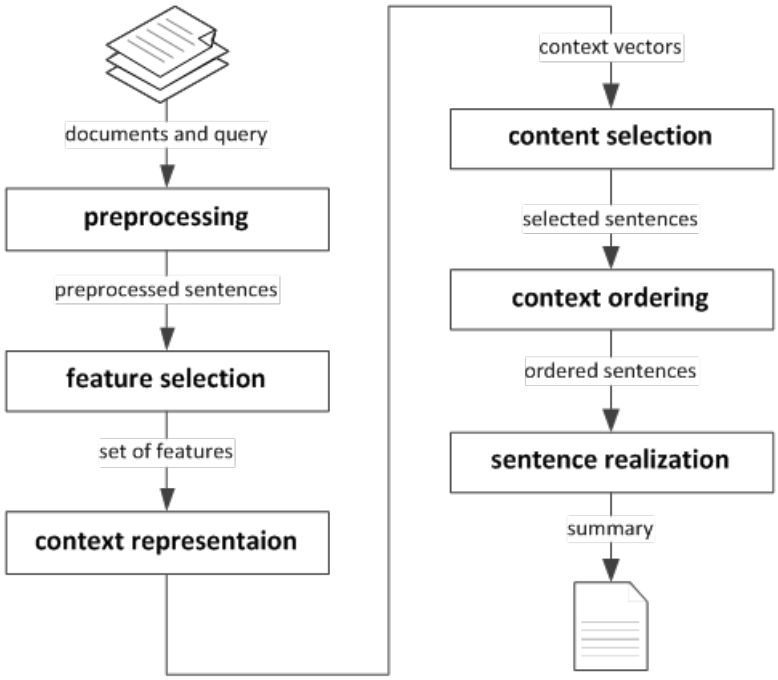

Figure 2.2: General extraction-based summarization workflow.

such as a phrase, a sentence or a paragraph. In the extraction-based automatic summarization task, a sentence is the primary type of context the summarizer deals with and therefore, a *sentence* is often referred to as a *context.* Depending on the task, the input to the summarizer may be either a single document or a collection of documents, henceforth referred to as *input documents* or *source documents*.

The workflow in the diagram includes six subtasks:

Preprocessing takes a raw text as an input and applies some basic routines to transform or eliminate textual elements that are not useful in further processing of textual data. In extraction-based summarization the output of this step is often a set of sentences where non-content words and punctuations are removed and content words are reduced to their base forms.

Feature selection identifies words, phrases or some other cues that appear in a preprocessed text and may serve as valuable characteristics of textual contexts. In automatic summarization these cues are often referred to as *features*, *topic terms* or *topic signatures*.

Context representation makes use of features obtained from the feature selection subtask to represent contexts in a form suitable for further processing. Contexts are often represented as *context vectors*.

Content selection identifies contexts that should be included in a summary. A variety of approaches can be used for this subtask. Some of them require to compute the similarities between textual contexts.

Context ordering arranges contexts selected in the content selection subtask to construct a coherent and readable text.

Sentence realization applies techniques that operate on a sub-sentence level with the goal to improve the readability and clarity of a text. Sentence simplification is one of these techniques.

The majority of summarization systems implement only a subset of the mentioned subtasks. For example, sentence realization techniques are rarely implemented. Often it is easier to ignore sentences that require modification than to apply sentence realization techniques, which can potentially generate ungrammatical sentences. Context ordering is also considered to be a hard subtask in multi-document summarization and is omitted in many implementations. The primary focus of the extraction-based summarization research is on content selection.

2.3 Preprocessing

The preprocessing steps commonly performed in automatic summarization are word stemming, stop words removal, text segmentation and query expansion. These routines are also quite common for other NLP tasks such as document retrieval, information extraction and machine translation.

2.3.1 Stemming

Stemming is the process of reducing the inflected forms of a word to a root form. It is commonly used in information retrieval tasks to reveal semantic similarity between different morphological variants of a word. For example, verbs “play”, “plays” and “played” all have the same root form “play” and thus might be represented by only one feature (i.e. term). This makes it possible to detect similarities between contexts that contain these words. In addition, word stemming reduces the overall number of features, making the data less sparse.

The simplest way to perform stemming is to use a comprehensive list of inflected forms for all words in a language. This may be quite unrealistic but has the advantage of being able to handle any type of inflection at any level of granularity without sophisticated methods.

Another way is to use a set of predefined language-specific rules to transform a word into its base form. Porter stemmer is, perhaps, the most well-known algorithm of this kind. It uses a set of

rewrite rules to eliminate suffixes. The problem with Porter stemmer is that it may sometimes be too rough, throwing away morphemes that are important for the word semantics. For example, words “organization” and “organ” are reduced to the same root form but they often have different senses and are used in different domains. It may be speculated, however, that for a small collection of related documents, stemming will less likely lead to a substantial loose of semantical information compared to a large collection of unrelated documents. In case of multi-document summarization the input is usually a relatively small collection of documents, related to a specific set of topics. Therefore, stemming is often applied. Another problem of pure rule-based stemmers is their inability to handle exceptional cases like past tense forms of irregular verbs, for example “broke” - “brake”. The viable solution is to combine the two described approaches by introducing the list of frequently used exceptions to a rule based stemmer.

2.3.2 Stop Words Removal

Stop words are high-frequency words of a language that don’t carry any particular information on their own. Such words are removed at the preprocessing phase to reduce the number of features. Closed class words such as pronouns, articles , prepositions and conjunctions are often included in stop words lists. In addition, some of the frequently used open class words such as auxiliary verbs are also included. Stop word lists of various granularity are freely available on the Web and often utilized in summarization. During the removal procedure all the words that appear in a list of stop words are removed from the source documents.

2.3.3 Query Expansion

In query-oriented summarization the content of a query provides valuable clues for deciding which parts of the documents are important. However, queries are often very short and it might be beneficial to expand them in order to obtain more clues. Several methods using a variety of external sources have been proposed for this task. The general idea behind many of them is to use external resources such as for example WordNet for identifying terms that are related to the terms in a query. External corpus might also be used for query expansion, especially when distributed context representations (see section 2.5 for details) are utilized.

2.3.4 Text Segmentation

Text segmentation is essential for extraction-based summarization. Segmentation divides an input document into separate textual contexts. Ideally, a context should express a relatively independent

and standalone piece of information. In extraction-based summarization, sentences are considered as contexts. Although it may seem that finding sentence boundaries is a trivial task, in fact, punctuational ambiguities make it rather hard. Many natural languages use periods to indicate sentence boundaries but periods can also signal abbreviations, initials or ordinal numbers. Extractive summarization requires high accuracy sentence boundary detection methods because even a single incorrectly separated incomplete sentence will greatly reduce the coherence and readability of a summary. Several approaches have been applied for this problem. Majority of the approaches are either rule-based or rely on machine learning to identify sentence boundaries.

Rule-based systems use hard-coded rules and predefined lists of lexical items such as abbreviations. For example, the sentence boundary detection rule-based system ER [5] makes use of regular expressions. The performance of the ER on English texts is quite impressive, even when compared to some recent machine learning systems. Although with a high-quality expert made rules, rule-based systems are able to achieve a very high performance, manual construction of rules for a large set of languages requires a substantial effort. Besides, some very exceptional cases are unlikely to be captured by human experts.

Systems that rely on supervised machine learning use classifiers (e.g decision tree, SVM, neural networks) to handle punctuational ambiguities. Classifiers are usually trained on a set of local contexts containing punctuation marks. Unlike rule-based systems, classifier-based systems are usually language independent and require a large training corpus. Early classifier-based text segmentation systems such as Satz [6] performed worse than rule-based systems but the recent ones such as the one used at TAC 2009 [7] can surpass rule-based systems with the error rate below 1%.

Both rule-based and supervised machine learning techniques are able to achieve high accuracy in sentence segmentation. However, the accuracy of both systems is dependent on the availability and proficiency of human experts involved in the construction of rules or creation of labeled corpus. This can be especially a tedious task when dealing with textual data from different application domains in many different languages. Unsupervised sentence segmentation methods are capable of accomplishing the task without involving human experts. An interesting algorithm for unsupervised detection of multilingual sentence boundaries have been proposed by Kiss and Strunk[8]. Their algorithm called PUNKT demonstrated an impressive performance on newspaper corpora in eleven languages achieving the mean accuracy of 98.74%. The approach discriminates between abbreviations and sentence boundaries using three properties of abbreviations. First, a period and the word preceding the period are usually tightly collocated. The degree of collocation can be determined by the log-likelihood ratio (described in detail in section 2.4). Second, abbreviations tend to be rather short, thereby the length serves as a good indication of an abbreviation. Third, many abbreviation contain word-internal periods. Combined together and quantified, these three

characteristics turned to be enough to detect abbreviations. It was also demonstrated that a similar approach can be successfully used for the detection of initials and ordinal numbers, thus providing a complete and accurate sentence segmentation solution.

2.4 Feature Selection

In NLP, textual contexts are often represented in terms of features. Sections 2.7.1 and 2.7.2 describe how features are used in extractive summarization: train a classifier, obtain centroid-based scores, measure similarity between contexts, etc. Most of the features used in summarization are lexical features such as individual words and phrases.

In statistical NLP a sequence of words is often referred to as an n-gram where n stands for the number of words in the sequence. N-grams provide a simple yet useful model where the next term can be predicted based on the n previous terms. NLP applications such as machine translation and speech recognition rely heavily on n-grams for language models. In tasks such as text categorization and automatic summarization n-grams are often used as features or topic signatures. For the purpose of extraction-based summarization, relatively short n-grams such as unigrams (single words) and bigrams (two-words phrases) are used.

Different words and phrases carry different amount of information. A variety of methods are applied to discard unimportant features and to weight the remaining ones in correspondence with their importance. The basic method mentioned previously in the report is stop words removal, which utilizes a predefined list of unimportant lexical features. More sophisticated weighting techniques such as TF-IDF and log-likelihood ratio use statistical information about distribution of words in a set of documents.

The TF-IDF weight is the product of two components; the term frequency TF and the inverse document frequency IDF. The TF determines the importance of a word for a given document, while the IDF indicates the importance of a word over the whole set of documents. A word that occurs often in a specific document but rarely in other documents is considered to be relevant for this document and, consequently, receives a high weight value. Formally, the TF-IDF weight for word w in document d is calculated as follows:

$$weight(w,d) = TF(w,d) \times IDF(w) \tag{2.1}$$

$$TF(w,d) = \frac{TC(w,d)}{|d|} \tag{2.2}$$

$$IDF(w) = log\frac{|D|}{|D(w)|+1} \tag{2.3}$$

where $TC(w,d)$ (term count) is the number of times word w occurs in document d, $|d|$ is the number of words in document d, $|D|$ is the number of documents in a collection of documents D, $|D(w)|$ is the number of documents in D that contain word w. The formulas above represent the classical account of the TF-IDF weighting scheme widely used in information retrieval and document classification. Automatic summarization is a little bit different from these tasks because its primary focus is on the short contexts like sentences rather then entire documents. Several modifications of the TF-IDF scheme are available that utilize frequencies of terms in sentences instead of documents. Most of these modifications use either the inverse sentence frequency, ISF, instead of the IDF or both IDF and ISF simultaneously [9, 10].

Although, in practice, TF-IDF works well for many applications including summarization, it has a somewhat weak statistical justification. The alternative is to use a statistical test to determine if a word w occurs significantly more often in a set of documents D that are relevant to a given topic than in a set of non-relevant documents $\tilde{D}$. In multi-document summarization a collection of input documents, which should be summarized, is used as D and a background corpus with non-relevant documents as $\tilde{D}$. A variety of statistical tests such as chi-square (χ^2) test and z-test can be used to identify the words that are important to the topic of documents in D. However, the likelihood ratio seems to be especially suitable for sparse data which is common in NLP tasks.

As defined by Ted Dunning the likelihood ratio for a hypothesis, denoted by λ, is the ratio of the maximum value of the likelihood function H over the subspace represented by the hypothesis to the maximum value of the likelihood function over the entire parameter space [11].

$$\lambda = \frac{\max_{\omega \in \Omega_0} H(\omega;k)}{\max_{\omega \in \Omega} H(\omega;k)} \tag{2.4}$$

where Ω_0 is the particular hypothesis being tested, Ω is the entire parameter space and k is the number of observations. The hypothesis we would like to test is whether a word w occurs significantly more often in relevant documents than in non-relevant documents and therefore, the presence of the word w in a document d indicates the relevancy of this document. The occurrence of a word w in a certain document can be modeled as a series of Bernoulli trials where the word w is matched with all the words contained in this document and the result of each match is either success or failure. Assuming the independence of trials the results follow the binomial distribution:

$$f(k;n,p) = \binom{n}{k} p^k (1-p)^{(n-k)} \tag{2.5}$$

where k is the number of successes trials in n trials (matches) and p is the probability of success in each trial. The joint distribution of the occurrence of a word w in independent sets of docu-

ments D and $\tilde{D}$ is the product of two binomial distributions with corresponding sets of underlying parameters:

$$H(k_1,k_2,n_1,n_2,p_1,p_2) = f(k_1,n_1,p_1)f(k_2,n_2,p_2) \tag{2.6}$$

where $f(k_1,n_1,p_1)$ and $f(k_2,n_2,p_2)$ are the binomial distributions of the occurrence of a word w in D and $\tilde{D}$ respectively, k_1 is the absolute frequency of a word w in D, k_2 is the absolute frequency of a word w in $\tilde{D}$, n_1 is the total number of words in D, n_2 is the total number of words in $\tilde{D}$, p_1 is the probability of a successful match of word w for documents in D and p_2 is the probability of a successful match of word w for documents in $\tilde{D}$.

If a word w occurs significantly more often in a set of relevant documents D than in $\tilde{D}$ $p_1 \gg p_2$, otherwise $p = p_1 = p_2$. The likelihood ratio for this test is formulated as follows:

$$\lambda = \frac{\max_p H(k_1,k_2,n_1,n_2,p,p)}{\max_{p_1,p_2} H(k_1,k_2,n_1,n_2,p_1,p_2)} \tag{2.7}$$

The maxima for the numerator is achieved with $p = \frac{k_1+k_2}{n_1+n_2}$ and for the denominator with $p_1 = \frac{k_1}{n_1}$, $p_2 = \frac{k_2}{n_2}$. It turns out that the confidence value for the $-2log\lambda$ can be obtained directly from the χ^2 distribution table because the log-likelihood ratio $-2log\lambda$ is asymptotically χ^2 distributed [12]. This value can be used as the threshold to filter out words that are not significant enough to determine the relevance of a document. Lin and Hovy [13] showed in their work that with $-2log\lambda > 10.8$ the log-likelihood weighting scheme outperforms the TF-IDF scheme when used for extraction-based summarization.

The use of single words as topic signatures is a mainstream approach in summarization. However, sometimes sequences of terms may be better indications of a topic. For example bigrams like “artificial intelligence”, “computer science”, “neural network” are obviously more informative and accurate than the separate words contained in them. However from all the bigrams selected from a text only very few will make sense. Bigrams such as “get new”, “usual means” and “undoubtedly think” are quite meaningless. The identification of meaningful bigrams is an important issue and have been thoroughly addressed in the context of text categorization. A nice overview of the papers and promising experimental results can be found in the work of Tan et al. [14]. Although automatic summarization is quite different from the text categorization task, the idea introduced earlier in the discussion of the log-likelihood ratio, where the set of documents is divided into relevant and non-relevant “categories” (i.e. subsets), makes it possible to borrow bigram extraction techniques from text categorization. The algorithm used by Tan et al. [14] relies on information gain (IG) to reveal the discriminatory power of a bigram. It can be shown that the log-likelihood ratio is proportional to the information gain [13] and therefore can also be used in the selection of informative bigrams.

An alternative approach is to use the log-likelihood ratio to measure the association between the words contained in a bigram rather than between a bigram and a set of relevant documents [15]. This way the meaningful bigrams can be extracted without using an additional background corpus. The general idea is to measure the deviation between the observed and the expected frequencies of a bigram with the initial assumption that words contained in the bigram are independent of each other. High deviation will indicate that the assumption is not true and that there is a certain dependency between the words, thus this bigram is meaningful.

The test uses the following contingency table:

	w_2	$\tilde{w}_2$
w_1	k_{11}	k_{12}
$\tilde{w}_1$	k_{21}	k_{22}

where w_1 and w_2 are the words contained in the bigram w_1w_2, $\tilde{w}_1$ and $\tilde{w}_2$ are words different from w_1 and w_2 respectively, k_{11}, k_{12}, k_{21} and k_{22} indicate the number of times the corresponding bigrams w_1w_2, $w_1\tilde{w}_2$, $\tilde{w}_1w_2$ and $\tilde{w}_1\tilde{w}_2$ are observed in a document or a collection of documents. The total number of bigrams is defined as $n = k_{11} + k_{12} + k_{21} + k_{22}$. Then the expected absolute frequencies (i.e. the number of times a bigram is expected to be observed) for bigrams w_1w_2, $w_1\tilde{w}_2$, $\tilde{w}_1w_2$ and $\tilde{w}_1\tilde{w}_2$, assuming the independence of the contained words, are calculated as follows:

$$
\begin{aligned}
e_{11} &= \frac{(k_{11}+k_{12})(k_{11}+k_{21})}{n} \\
e_{12} &= \frac{(k_{11}+k_{12})(k_{12}+k_{22})}{n} \\
e_{21} &= \frac{(k_{21}+k_{22})(k_{11}+k_{21})}{n} \\
e_{22} &= \frac{(k_{21}+k_{22})(k_{12}+k_{22})}{n}
\end{aligned}
$$

Finally the log-likelihood ratio is computed:

$$
2\sum_{ij} k_{ij} log \frac{k_{ij}}{e_{ij}} \tag{2.8}
$$

The threshold can be obtained from χ^2 distribution table. Bigrams with the value higher than this threshold are considered as important features.

The feature space that consists of bigrams with high weights will likely to be very sparse. Therefore, a combination of high-weighted bigrams and unigrams is probably the reasonable solution for NLP tasks including summarization.

2.5 Context Representation

This section provides an overview of textual representation models used in various NLP tasks including automatic summarization. These models are used to represent textual data in the way that facilitates intelligent processing of this data.

In general, processing of the data on a computer requires data structures and algorithms. In AI these concepts are typically referred to as representation and inference models respectively. On the "register-transfer" level [16] textual data like any other information stored in computer memory is represented as a sequence of bits. On the "symbol level", bits make up letters, letters are combined in words, words in sentences and so on. However, in many NLP tasks the primary interest is the meaning of a text rather than the syntactical from. Unfortunately, extraction and representation of meaning contained in a text is a challenging problem, far from being solved. The alternative is representations that are still at the symbol level but are able to reveal at least some semantic information that makes them suitable for intelligent processing. Models described in this section belong to this class of representations.

The most widely used textual representation model is, without doubt, the vector space (VS) model. In this model, a document, a query, a sentence or any other textual context is represented as a vector of features that indicate the presence of a specific term or sequence of terms in the context. The vector space model can be characterized as the bag-of-words model where the order of terms is ignored. Feature values are often based on term frequencies and frequency distribution factors. Some of the features and weighting schemes was described in section 2.4. Considering the compositionality of language semantics, there is a strong connection between the meaning of a context and the meaning of the high frequency terms contained in this context. The vector space model relies on this assumption and makes this connection explicit. Although features themselves are not understood by a computer, their values provide relative semantic clues about the context. For example, a Wikipedia article about anteaters contains terms "termits", "mammal", "animal", "food", "economy", "finance", "crisis" with absolute frequencies 4, 3, 5, 2, 0, 0, 0 respectively. Using the vector space model with absolute frequencies of this terms as feature values, this document can be represented by a vector $\vec{d} = (4,3,5,2,0,0,0)$. This vector clearly reveals the fact that the article is semantically related to senses of the words "termits", "mammal", "animal", "food" but not to "economy", "finance" and "crisis" although the semantics of individual terms is not understood by a computer. From the human perspective, it is obvious that anteaters have nothing to do with a financial crisis in the world's economy. The model could capture this intuition without deep understanding of the knowledge contained in the article. Representation of textual contexts as vectors makes it easy to apply techniques from geometry and linear algebra. Vectors can be collected in matrices for large-scale processing. The generalization done by the VS model facili-

tates the process of finding similarities between contexts (i.e. context vectors). This makes the VS model suitable for a variety of applications such as summarization, information retrieval and text categorization where the contexts need to be compared to each other. At the same time, this model throws away some information such as a term order that is important for tasks such as detection of textual entailment.

Although the traditional VS model proved to perform well in many NLP tasks, it has some downsides that should be addressed. The major one is the sparseness of data. According to Zipf's law, frequency of a word in some corpus is inversely proportional to its frequency-based rank. The classical version of Zipf's law for the frequency of words in the English language can be formulated as follows:

$$f_k \propto \frac{1}{k} \tag{2.9}$$

where k is the frequency-based rank of a word and f_k is the frequency of this word in the corpus. Thus, the most frequent word in the language will occur approximately twice as often as the second frequent word. For a large set of documents the number of unique words is huge but most of these words occur only very few times. Consequently, short contexts such as sentences represented as vectors will contain very few non-zero values, that is, the contexts vectors will be very sparse. The main problem with this is that similarity-based algorithms perform quite poorly on sparse data because context vectors are nearly orthogonal. Thus, the similarities between most contexts will be zero. This degrades the performance of similarity-based algorithms such as clustering and centrality-based methods. In some summarization systems relatively few features are selected in advance and the contexts that don't contain any of them are ignored. This way the sparseness of data becomes less crucial. There exist several other techniques for reducing the dimensionality of the vector space without omitting lexical features that might be useful. These are often referred to as dimensionality reduction or feature/topic extraction techniques. They represent the meaning of terms and textual contexts in a distributed way based on the co-occurrence relations between the terms.

Latent Semantic Analysis (LSA) [17] is, perhaps, the most well-known and successful one among the dimensionality reduction techniques. It uses a matrix factorization method called *singular value decomposition* (SVD) to approximate the initial term-context matrix by a matrix of a much smaller size. SVD decomposes a term-context matrix X of size $t \times c$ into three matrices:

$$U, S, V^T = SVD(X) \tag{2.10}$$

$$X = USV^T \tag{2.11}$$

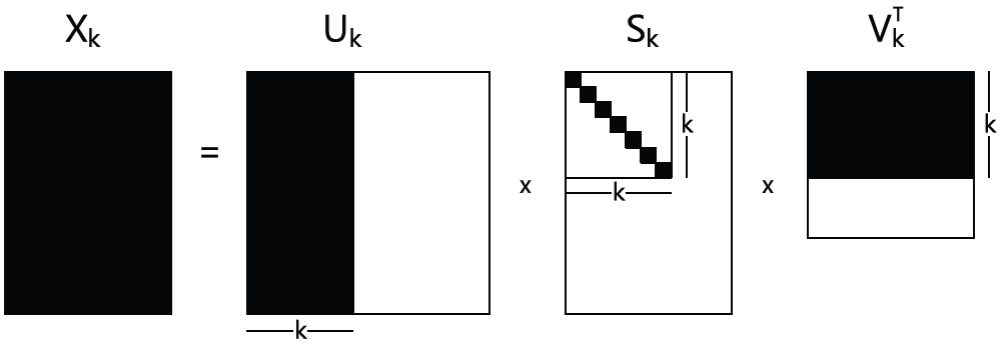

Figure 2.3: Truncation of matrices obtained through SVD.

where

- U is a matrix of size $t \times t$. It consists of orthogonal columns that are left eigenvectors of XX^T. Rows in this matrix represent the meaning of terms based on their cooccurrences with other terms.
- S is a diagonal matrix of size $t \times c$ with the values referred to as singular values that indicate the importance of each dimension.
- V is a matrix of size $c \times c$. It consists of orthogonal columns that are right eigenvectors of X^TX matrix. Rows in this matrix represent the meaning of contexts based on other contexts that they share terms with.

These matrices are then truncated to k dimensions as shown in figure 2.3, leading to elimination of the least significant (noisy) dimensions. Number of dimensions k is usually determined empirically depending on the initial size of the matrix X and the task at hand. The truncated matrices are used either separately or in combination with each other. For example, rows from V_k or V_kS are often utilized to measure similarities between contexts. Approximation of the original term-context matrix X_k can be obtained by multiplying the reduced matrices.

Instead of SVD other methods such as principal component analysis and independent component analysis are also used. The general idea behind these methods is to detect latent relations between contexts and features based on co-occurrence patterns. Related features are collapsed as the result of orthogonalization process and values of other features are smoothed to produce a smaller but dense matrix. Merging of term features that often appear in the same contexts has a significant semantic effect. To some extend, the reduced set of features relates to senses of words rather than their forms. This effect can be explained by *distributional hypothesis* [18], which states that

words that occur in the same context tend to have similar meanings. LSA have been applied to a variety of NLP tasks including information retrieval, document categorization and information filtering. In the research carried out by Steinberger et al. [19, 20] LSA was successfully used for single-document and multi-document summarization.

Despite the fact that in general LSA performs well, there are several reasons why sometimes it can't be utilized effectively. The first reason is the computational costs, with regard to both memory consumption and execution time. Even though efficient SVD algorithms exist, the size of the input matrix constructed from a large corpus such as a collection of web-pages obtained from the Internet can make LSA practically infeasible. The situation becomes even worse when new contexts are added to the collection incrementally. The whole procedure should be performed from scratch each time a new context arrives, so no easy update of the model is possible. Another problem is the interpretation of the results. In extraction-based summarization a large number of term features can be transformed to reduced conceptual space. No explicit relation between the original feature space and the reduced one is provided by SVD, which makes it hard to interpret the meaning of new dimensions. After reduction, it becomes possible to measure the similarity between contexts effectively, albeit without understanding what makes them similar.

Another dimensionality reduction technique, which is substantially different from LSA, is Random Indexing (RI) [21]. Based on the previously mentioned distributional hypothesis, Random Indexing represents the meaning of a lexical feature through the contexts in which it occurs. However, unlike LSA, RI utilizes a simple iterative procedure, which makes it suitable for very large amounts of data. RI is motivated by the observation that in a high-dimensional feature space many context-vectors are nearly orthogonal. With this assumption, the straightforward way to represent a context is to use an unary vector of the same size as the total number of contexts where all the values are zero except the one that corresponds to the index of a particular context in the collection. For example, the third and the fifth context in a collection of five contexts can be represented by the following index vectors:

$$\vec{c}_3 = (0,0,1,0,0)$$

$$\vec{c}_5 = (0,0,0,0,1)$$

Then, the meaning of a term is represented as the sum of vectors of the contexts in which this term occurs. That is, the term that occurs in contexts three and five is represented by the following term vector:

$$\vec{t} = \vec{c}_3 + \vec{c}_5 = (0,0,1,0,1)$$

Term vectors can be collected in a matrix $F_{t\times c}$ which is an approximation of the term-context matrix. Obviously the sizes of the original term-context matrix and the approximation $F_{t\times c}$ are

identical. However, the dimensionality of $F_{t\times c}$ can be reduced by using randomly sampled index vectors of smaller size instead of unary ones, which will still preserve the approximate orthogonality of the index-vectors. The Johnson-Lindenstrauss lemma states that the distances between the points projected into a randomly selected subspace of sufficient high-dimensionality will be nearly the same as in the original space. Thus, the projection can be done by multiplying matrix $F_{t\times c}$ by random matrix $R_{c\times k}$:

$$F_{t\times c}R_{c\times k} = F'_{t\times k} \tag{2.12}$$

where $k \ll c$ and $F'_{t\times k}$ is a matrix of which rows represent the meanings of terms in a distributed way. The random matrix R is usually sampled from normal distribution or some simple zero mean distributions. The obtained representation of terms in $F'_{t\times k}$ can then be used to represent a context by summing up the index-vectors of terms contained in the context. Like LSA, RI can also be used for a variety of tasks where the efficient computation of similarity between words and textual contexts is required. There have been an attempt to apply RI for single-document summarization [22]. Until now no experimental research have been done using RI for multi-document summarization but in theory, it should be able to provide better results than local representations by revealing latent similarities between contexts.

2.6 Similarity Measures

Textual similarity is a complex concept that can be defined as the semantic relatedness of two textual contexts. Two contexts are considered similar if they focus on the same or related concepts, actors or actions. Measuring the similarity between textual contexts is an intermediate step for many NLP applications. Most of the research regarding text similarity have been done by IR community, where assignment of similarity between a query and a document is a core task. In automatic summarization, similarity metrics are used for centrality-based context selection and in identification of redundant contexts. In general, similarity measures are either corpus-based or knowledge-based. Both of them have been used in extractive summarization. Corpus-based measures use term frequencies observed in a corpus to relate contexts to each other, while knowledge-based methods utilize predefined semantic relations between terms obtained from lexical resources.

Corpus-based metrics often make use of the vector representation described in section 2.5. An obvious way to measure the similarity between two vectors in a multidimensional space is to use either euclidean or angle distances. The similarity based on euclidean distance for two contexts represented as vectors $\vec{p}$ and $\vec{q}$ is calculated as follows:

$$similarity(\vec{p},\vec{q}) = \frac{1}{1+\sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}} \tag{2.13}$$

The cosine similarity metric based on the angle between two vectors is perhaps the most widely used similarity metric in NLP. It is calculated as follows:

$$similarity(\vec{p},\vec{q}) = \frac{\vec{p}\cdot\vec{q}}{|\vec{p}||\vec{q}|} \tag{2.14}$$

In extraction-based summarization, sentences are often used as contexts. Sentences in most natural languages tend not to have more than one occurrence of the same content word. Therefore, context vectors usually contain only ones and zeros. In this respect boolean set-based similarity metrics can be used. On of these is the Jaccard similarity coefficient, defined as the size of intersection divided by the size of the union of the two sets:

$$similarity(P,Q) = \frac{|P\cap Q|}{|P\cup Q|} \tag{2.15}$$

where P and Q are the sets corresponding to boolean vectors $\vec{p}$ and $\vec{q}$ respectively. All these metrics will perform quite poorly with nearly orthogonal vectors and that is why distributed representations such as RI or LSA are often used.

Knowledge-based approaches check the relatedness between terms occurring in different contexts through semantic links. Semantic links are obtained from lexical resources such as WordNet, VerbOcean and Roget's Thesaurus. Different resources provide different types of semantic relations. For example, the WordNet contains a synset hierarchy including glossies, meronyms and hypernyms. Most often synonymous relations and glossies are used to determine the semantic overlap between contexts. Basically, the similarity score increases if two words from different contexts are in the same synset or one of them occurs in the definition of another one. This approach have been used in the *concept link method* applied for extraction-based summarization where the similarity between sentences is determined by matching WordNet entries of open-class terms contained in the sentences [23]. A similar approach to summarization have also been employed using the Roget's Thesaurus [24].

Another interesting knowledge-based similarity approach is using encyclopedic knowledge. The popular encyclopedic online resource Wikipedia contains a tremendous amount of target specific information. Several different summarization approaches have been recently proposed utilizing Wikipedia for query-expansion, feature selection or similarity assignment [25, 26]. Wikipedia provides articles devoted to a variety of concepts. The first step is to obtain Wikipedia articles for concepts contained in the compared contexts. This process usually involves disambiguation[1] of the concepts, which is done by matching the input document with candidate Wikipedia articles. The

[1]For example a Wikipedia disambiguation page for a word "apple" refers to 3 different companies, 3 movies, 2 music bands and many other entities. The page is available at http://en.wikipedia.org/wiki/Apple_(disambiguation)

articles for disambiguated concepts from one context are than matched with articles for concepts in another context. The overlap between them is used as the similarity measure for the contexts. It can be quite costly in terms of performance to utilize entire Wikipedia articles for matching, so often only first sentences or sections are used. Wikipedia articles are interconnected with links that might also be utilized for similarity assignment.

2.7 Content Selection

Selection of sentences is the core process in extractive summarization. The goal of the selection procedure is to identify a set of sentences that contain important information. Three criteria are optimized when selecting the sentences: relevance, redundancy and length. Relevance determines the importance of the information contained in a summary with respect to the topics covered in the source documents or a query in case of query-focused summarization. Redundancy measures the information overlap between the sentences selected for the summary. Given a restricted summary length, summarization systems try to maximize the relevance while minimizing the redundancy. The task of content selection is to identify which sentences in the source documents are worth taking into a summary. The problem can be handled either in a supervised or unsupervised manner.

2.7.1 Supervised Methods

Supervised techniques use a classifier trained on a set of documents coupled with corresponding extracts. Extracts make it possible to label sentences in the documents with a binary value: 1 - a sentence is included in the extract, 0 - a sentence is not included in the extract. In addition each sentence should be represented by a feature vector. A feature vector is constructed using a predefined set of features that should provide enough information to determine whether a particular sentence is worth including in a summary. Some of the commonly used types of features are:

- cue phrases and topic terms, for example terms with high TF-IDF or log-likelihood weights
- position of a sentence in a document, for example which section of a document a sentence occurs in (the first and the last sentences tend to be very informative)
- centrality of a sentence, for example a similarity between a sentence and other sentences in a document
- length of a sentence, for example the number of open-class words (i.e. nouns, main verbs, adjectives, adverbs) in a sentence

First, a classifier is trained on a set of feature vectors with corresponding labels (i.e. included, not-included in the extract). Then the classifier is used to estimate labels for unseen sentences. A variety of different classifiers can be used for this task. For example a simple probabilistic classifier, naive Bayes, calculates the conditional probability $P(worthy(s)|f_1,...,f_n)$ of a sentence s represented as a feature vector $(f_1,...,f_n)$ to be worthy of inclusion in the extract. Sentences with a probability higher than some predefined threshold are selected. Supervised approach are rarely used on their own but mainly as a way to combine all the available evidence obtained by other approaches.

The major problem with the supervised approach is the availability of training data. Manually created summaries are mostly abstracts rather than extracts. Consequently, sentences in the source documents will be different from sentences in the abstract and should be aligned in order to obtain the labels. Basic alignment algorithms use longest common subsequences and minimum edit distance to obtain the similarity between sentences. Sentences from the source documents that closely match sentences from the abstract are assigned the label 1 and others get 0. With an alignment algorithm in place one can utilize a large collections of documents with abstracts such as scientific papers to train the classifier. Although possible, the supervised approach is rarely used for multi-document summarization because of the unavailability of training sets that contain summaries of several documents.

2.7.2 Unsupervised Methods

The unsupervised content selection methods identify salient sentences without training on a labeled set of documents. Most of the unsupervised summarization algorithms are either *centroid-based* or *centrality-based*.

The general idea behind centroid-based algorithms is to select sentences that contain informative words, also refereed to as *topic signatures*. Stop words like articles and pronouns are usually ignored. The informativeness of the remaining words is calculated using popular weighting schemes such as TF-IDF or log-likelihood ratio. Then, the score for a sentence is calculated as follows:

$$score(S) = \frac{1}{|S|} \sum_{w \in S} weight(w) \tag{2.16}$$

where w is the word that appears in a sentence S and $|S|$ is the number of words in the sentence S.

A somewhat different model is used in the centrality-based algorithms where sentences are ranked based on their similarity with other sentences in the same document or in the set of input documents. The intuition behind this approach is that the core information will be repeated or referred

to in many sentences, thus the sentences that are similar to many other sentences are likely to contain this information and should be included in the extract. The algorithm requires computing the similarities between sentences in the input documents. When the similarity between each pair of sentences is available, a centrality algorithm can be applied to obtain a score for each sentence. The most straightforward way to compute this score for a sentence S is to take the average of the similarities between S and all the other sentences:

$$score(S) = \frac{1}{|D|} \sum_{U \in D} similarity(S,U) \tag{2.17}$$

where U is a sentence other than S from a set of sentences D, $|D|$ is the total number of sentences in the input documents excluding S. Graph theory provides much more sophisticated measures of centrality than the one mentioned above. Many of the measures will be thoroughly discussed in section 3.

The algorithms described above rely on superficial features ignoring higher-level semantic information such as semantic relations between terms or coherence relations between utterances in the discourse. Although acquisition of this information is a complex task on its own, at least two summarization approaches are known to make use of it. The first approach is the so called *event-based summarization*. In event-based summarization [27, 28, 29], sentences are selected based on the importance of events they describe. Events are often defined as verbs, action nouns and named entities and are identified using the feature weighting schemes described in section 4.3. Events are connected through semantic relations and lexical resources such as WordNet or VerbOcean may be used to obtain such relations. These connections are exploited to identify the most central events using a graph-based centrality algorithm. Finally, the sentences that refer to salient events are included in the extract.

Another way to go beyond term-level features is to consider *coherence relations* between sentences. Coherent relations are usually defined as semantical relations between discourse units, such as explanation, contrast, elaboration, evidence, etc. Several theories of coherent relations with different sets of relations exist. The one that has been considered for the purpose of summarization is the Rhetorical Structure Theory[30]. The suggested approach utilizes a discourse parser to construct a rhetorical tree from the input document. The rhetorical tree organizes the discourse units (sentences, subordinate clauses) into a hierarchical structure through identification of semantic relations. The salience of each unit is then determined based on its position in the tree, its role and the roles of its ancestors in the relations. The discourse units with high salience scores are included in the summary. Unfortunately, the identification of coherence relations is still an open research problem and existing approaches are unable to provide very accurate results. The known methods applied for the task rely on cue phrases such as “because”, “but”, “although”, “for

example", "and". These phrases may signal a discourse structure but can be quite ambiguous in regards to which coherence relations they exactly indicate. Quite often, however, no cue phrases are available at all. Then, supervised machine learning techniques are used to train a classifier to distinguish between relations based on contextual features [31].

2.7.3 Redundancy Removal

Redundancy is a major issue in multi-document summarization where several documents on the same topic may have a substantial information overlap. Then, the selection of the most relevant sentences will yield a set of sentences with redundant information. Extract that consists of relevant but very similar sentences is not good. The joint optimization of both relevancy and redundancy is a complex task because properties of individual sentences are dependent on other sentences included in the summary. Some of the earlier multi-document summarization approaches handle these optimizations separately. For example, clustering is used to obtain groups of similar sentences and then the most authoritative contexts from each group are selected [32]. This way the clustering step minimizes redundancy and the selection of authoritative contexts from each cluster maximizes relevance.

A more explicit way of joint optimization of the criteria is to include both redundancy and relevancy in computation of scores when selecting sentences. One of the first models that was able to do so is the maximal marginal relevance (MMR) model proposed by Carbonell et al. [33], where relevance and redundancy are combined into one function. The context selection procedure selects the context with the highest rank at a time and then rescores all the remaining contexts. The rescoring procedure reduces the scores of contexts that are similar to the already selected contexts thus reducing the redundancy. Formally the MMR score for a sentence s is calculated as follows:

$$score(s) = relevance(s) - \lambda \max_{s_i \in summary} similarity(s, s_i) \tag{2.18}$$

where λ is a constant that controls the information diversity of a summary. This way the optimization happens in a greedy fashion and the global maximum is not guaranteed. More recent approaches [34] apply dynamic programming and integer linear programming to reach the global maximum of the score function.

A simple but elegant algorithm for content selection that handles redundancy have been used in the SumBasic summarization system [35]. The algorithm utilizes word frequencies as the indicator of an informative sentence. The intuition behind the proposed method is that words that occur frequently in the input documents tend to occur frequently in a summary. According to this intuition, relative word frequencies are used as probabilities of a word in an input document to occur in the

summary. Sentences with high-frequency words are included in the summary. The redundancy is avoided by decreasing the probability for words contained in the already selected sentences. For example, the initial probability of a word is squared if the word appears in the summary once and cubed if twice.

Another way to deal with dependencies between contexts is to represent them explicitly in a graph. In most graph-based approaches the relations between contexts are defined by their similarity although other types of relations such as coherence are possible. Most of the graph-based summarization approaches optimize only the relevance criteria [36, 37] and only few optimize both relevance and redundancy [38].

2.8 Context Ordering

One of the challenges in extraction-based summarization is how to organize the selected contexts into a coherent summary. Context ordering techniques are supposed to handle this problem.

In single document summarization the original document already provides an ordering of contexts which that can be preserved in a summary. However, it was observed that even in single document summaries manually created by professional summarizers the selected sentences do not always follow the original ordering [39]. Obviously in multi-document summarization, when contexts are coming from several documents, the original ordering can't be used. Documents often contain different pieces of information and even with the large overlap the ordering can change from one document to another. Several approaches for context ordering are discussed below.

One approach for ordering the contexts from several documents is to use chronological information like a publishing date or time information contained in a document [32, 40]. Unfortunately, not always this information is available or is appropriate for arranging documents and contexts. For example, contexts from two documents published in the same date can't be ordered using this approach.

Majority Ordering [41] approach tries to provide a coherent arrangement of contexts by grouping them into topics. Clustering methods are often used for unsupervised grouping of contexts. One problem with this approach is to decide upon the number of clusters. Another one is how to order contexts within a cluster and the clusters relative to each other.

The probabilistic model for text structuring [42] relies on machine learning to obtain conditional probabilities of sentence pairs from a training corpus. Then these probabilities are used to estimate the global optimal ordering of contexts by a simple greedy algorithm.

The adjacency model [43] uses context connectivity from the original document. The summary is constructed by organizing sentences in a way that two neighbor sentences are highly connected with each other. The proposed method chooses the next sentence with the highest connectivity to the previous one, so that the choice of the sentence influences the ordering of sentences after it.

3.7 Sentence Realization

Sentence realization techniques modify candidate sentences or the sentences that have already been included in a summary in order to make the summary more concise as well as to improve the clarity and readability of the summary. In this section a brief overview of the sentence realization techniques is provided with regard to extraction-based automatic summarization.

One of the limitations of the traditional extraction-based summarization algorithms is their inability to handle contexts smaller than a sentence. Even with perfect sentence selection and ordering techniques the quality of the constructed summaries can by substantially improved by using sub-sentence level contexts. Obviously, selection and ordering of textual pieces at the sub-sentence level may lead to a grammatically incorrect and incoherent summary. Current text generation models are simply not good enough for this task. The reasonable solution is to modify existing sentences instead of generating the new ones. This way correct grammatical structures and coherence can be ensured while optimizing relevance and redundancy. There are two main techniques used in summarization that have been applied for this task: *sentence simplification* and *sentence fusion*. Both modify sentences in order to reduce redundancy and to exclude unimportant phrases. A third technique used in summarization is co-reference resolution. It modifies sentences with the goal of reducing semantic uncertainty in regard to unclear co-references.

Currently, a relatively small number of summarization systems make use of the mentioned techniques, primarily, because of their complexity. However, there is a trend in the field of automatic summarization towards developing more robust sentence realization algorithms that may substantially improve the performance of modern summarization systems.

2.8.1 Co-reference Resolution

Sentences selected for the summary during the content selection procedure may be difficult to interpret, even with the assumption that these sentences cover similar topics and provide a good understanding of the global context. An obvious example is a sentence that refers to some concept introduced in sentences that are not included in the summary. It turns out that this pattern is very common, especially in news articles. A good summarization system is required to deal with such

sentences, otherwise the produced summary may be hard to understand or, even worse, it may be misunderstood. Co-reference resolution techniques aim to handle this problem. Consider the following example:

> One proposal is to reduce the loan size on which interest may be deducted. Doing so could reduce the amount of tax revenue, and would make the deduction less skewed toward higher income households.

If the summary generated by a summarizer includes only the second sentence from this example, and the preceding sentence is rather different than the first sentence from the example, the "doing so" in the second sentence may be interpreted in a totally different way than it is intended in the example. Co-reference resolution resolves the reference by finding the source among the terms preceding the reference term and replacing the reference with the source term. Unfortunately, the existing co-reference resolution methods are not very accurate. Taking into consideration that even a few incorrectly resolved references can make a short summary very confusing, using these methods for summarization is not a good option. The example provided above clearly demonstrates the complexity of the task. It is not enough just to detect that the second sentence refers to the concept of "reduce the loan size". The substitution of the reference may require adjustment of the grammatical structure:

> Reduction of the loan size could reduce the amount of tax revenue, and would make the deduction less skewed toward higher income households.

The best alternative here is probably to include both sentences. Another alternative is to avoid sentences with references completely, which is quite restrictive because it would possibly ignore a lot of informative sentences. A third alternative is to include sentences with references that can be resolved within the same sentence. For example:

> Google says it is not responsible for how third-party apps perform on its phones.

Detection of such sentences is much simpler than the general task of reference resolution and can be done by a trained classifier. In general, the task of reference resolution is an active area of research and automatic summarization is one of the applications which could greatly benefit from the use of high accuracy resolution methods [24].

2.8.2 Sentence Simplification

The goal of the sentence simplification (also known as a sentence compression) is to make sentences more concise by removing redundancy and uninformative parts. This way more of relevant information can be included in the summary. Sentence simplification can be applied either during the preprocessing step, explained in section 2.3, or postproccessing, when the sentences that have been already included in a summary are simplified. Simplification rules applied in the post processing phase are usually quite conservative to avoid oversimplification. The alternative is to apply simplification techniques during preprocessing. This way simplified versions of a sentence are presented to a content selection component that decides which of them should be selected. The original sentence can be also included. Combined with the content selection techniques that avoid redundancy this approach proved to be more successful than the simplification done during post-processing. Assuming that the content selection will assign low scores to oversimplified sentences, more aggressive simplification techniques can be applied. In general, use of sentence simplification before content selection step gives better results [44].

Simplification techniques usually rely on syntactic rules for trimming the phrases that don't convey the core meaning of a sentence. Common syntactic patterns used for the identification of such phrases are:

Gerund clause: An oil rig, burning in the Gulf of Mexico for more than a day, has sunk.

Non-restrictive relative clause: Even if you had a really strong cup of tea or coffee, which is quite hard to make, you would still have a net gain of fluid.

Intra-sentential_attribution: Tokyo and Okinawa are the only prefectures that are expected to see their populations rise during that period, the report said.

Lead adverbs: Obviously, they have had sponsorship from some big companies.

In order to apply these rules, sentences are represented as parse trees. Both full-fledged and shallow parsers have been used. Then, certain nodes according to the patterns are eliminated. Additional clues and conditions are often used to handle special cases. Finally, minor corrections in punctuation and capitalization are introduced before making simplified sentences available for a content selection algorithm. In general, summarization systems that utilize sentence simplification demonstrate impressive performance when combined with a content selection mechanism that minimizes redundancy [45, 46].

2.8.3 Sentence Fusion

Sentence fusion techniques combine several related sentences into one sentence. The motivation behind this type of sentence realization approach is the observation that sentences in manually constructed summaries tend to include information from two or more sentences in the source documents. The ability to do so in automatic summarization is a step towards abstraction-based summarization. An interesting implementation of sentence fusion in the context of automatic summarization was proposed by Barzily and McKeown [47]. Their algorithm consists of three phases. During the first phase the redundant information contained in the sentences is used to identify the overlap between them. Technically, this is done by representing sentences as dependency trees, which is a parse tree with some grammatical structure and shallow semantic information. Then the trees are aligned based on lexical and structural similarities between sub-trees. Synsets from WordNet is used in order to handle paraphrasing. The second step is to combine sub-trees from different sentence trees. This is done by replacing an overlapping sub-tree from one sentence with a sub-tree from another sentence. The decision on which sub-trees should be replaced is based on the alignment done in the previous phase. The final step is generation of sentences from dependency trees. A statistical linearization algorithm is used to select the best phrasing, placement of attached phrases and optimal ordering. The described approach demonstrates promising results and reveals a need for probabilistic models that will be able to discriminate between fluent and ill-formed sentences. Such algorithms will allow generation of several alternative versions of a fused sentence and then select the most fluent one.

2.9 Evaluation

Evaluation of summarization systems is a non-trivial task. Unlike document retrieval and text categorization where the robust evaluation metrics such as recall and precision are clearly defined, the evaluation of extracts and especially abstracts requires substantially more advanced techniques. In general, the evaluation methods are usually classified into *extrinsic* and *intrinsic* methods.

The extrinsic methods evaluate summaries based on a specific task. For example, subjects (humans) are given either automatically generated summaries, summaries produced by humans, or the original documents and then are asked to answer a set of questions about the content of the original documents. The answers of the subjects that were given only the automatically generated summaries are compared to the subjects having the manually created summaries and those who had the original documents. Based on this comparison the quality of the automatically generated summaries is determined.

Intrinsic evaluation is task-independent. In intrinsic evaluation, automatically generated summaries are directly compared to the manually constructed ones, referred to as *reference summaries*, and the overlap between them is calculated. The assumption is that a good automatically generated summary should contain the same information as the reference summaries, so a summary with a larger overlap is the better one. Unfortunately measuring the information overlap between summaries is far from a trivial task even when done manually especially when this overlap should be precisely quantified. In addition to the content overlap the fluency of a summary may be evaluated.

The mainstream approach in the field is to use intrinsic methods, both manual and automatic. In this section the most popular evaluation methods are described.

2.9.1 Manual Evaluation in TAC

TAC uses the combination of intrinsic methods for the evaluation of summarization systems that participate in the competition. The manual methods are used to evaluate summaries for content relevance, readability/fluency and overall responsiveness. Content is evaluated using the so called Pyramid Method [48], which assigns a score to a summary based on the number of units of meaning shared between the automatically generated and human summaries. The units of meaning, referred to as Summary Content Units (SCU) are manually extracted from the human-constructed summaries and roughly correspond to informative statements like the underlined words in the following text snippet:

> Heathrow, to the west of London, is one of the world's major international airports and has seen no activity the past five days.

Each SCU is weighted considering its relevance and informativeness. Then human assessors identify SCUs in candidate summaries. The final score is the ratio between the sum of unique SCU weights that appear in the candidate summary and the sum of SCU weights in the optimal summary with the same number of SCUs. The Pyramid Method is a recall-oriented measure which calculates the proportion of the highly weighted SCUs found in a candidate summary. The readability/fluency and overall responsiveness evaluation are done manually by assessors that assign scores (very poor, poor, barely acceptable, good, very good) to summaries for each of these criteria. The readability score reflects the fluency of the summary independent of whether it contains any relevant information or not. Some of the factors considered to be important for readability are coherence, structure, non-redundancy, grammaticality and preferential clarity. In the evaluation of the overall responsiveness both readability and relevance are considered by the assessors.

2.9.2 ROUGE

Automatic evaluation methods are not used in TAC, probably because of the quite poor correlation between the results obtained from manual evaluation and existing automatic techniques. Nevertheless, at least one of the automatic methods called ROUGE is used by many research groups in the field. ROUGE, Recall-Oriented Understudy for Gisting Evaluation [49] calculates scores for a candidate summary based on the n-gram overlap between the candidate and reference summaries. ROUGE includes several metrics such as ROUGE-1, ROUGE-2 and so on, each corresponds to the size of the n-grams used in the evaluation. ROUGE-N scores are computed as follows:

$$ROUGE\text{-}N_{recall} = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{n\text{-}gram \in S} count_{match}(n\text{-}gram)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{n\text{-}gram \in S} count(n\text{-}gram)} \quad (2.19)$$

$$ROUGE\text{-}N_{precision} = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{n\text{-}gram \in S} count_{match}(n\text{-}gram)}{\sum_{n\text{-}gram \in CandidateSummary} count(n\text{-}gram)} \quad (2.20)$$

$$ROUGE\text{-}N_{F\text{-}score} = \frac{2 \times ROUGE\text{-}N_{recall} \times ROUGE\text{-}N_{precision}}{ROUGE\text{-}N_{recall} + ROUGE\text{-}N_{precision}} \quad (2.21)$$

where $count_{match}(n\text{-}gram)$ is the number of times an *n-gram* from a reference summaries appears in a candidate summary and $count(n\text{-}gram)$ is the number of times *n-gram* appears in a candidate or a reference summary. Thus, the recall measure calculates the proportion of n-grams from reference summaries that occur in a candidate summary and the precision calculates the proportion of n-grams from a candidate summary that occur in reference summaries. F-score combines recall and precision into one metric.

The ROUGE package includes other metrics besides ROUGE-N. ROUGE-L and ROUGE-W metrics measure the longest common subsequence (LCS) shared by a candidate and reference summaries. A subsequence is defined as a sequence obtained from another sequence by removing some of its elements without changing the order of the remaining ones. For example $[a,b,c]$ is the subsequence of $[a,e,f,b,d,c]$. In ROUGE the LCSes are first obtained on a sentence level and then aggregated for the whole summary. Unlike ROUGE-L, the ROUGE-W metric assigns weights to subsequence matches based on a spatial distribution of terms in a candidate sequence. For example in the following sequences:

$$\begin{aligned} Reference &: [\underline{a},\underline{b},\underline{c},d,e,f] \\ Candidate_1 &: [k,\underline{a},\underline{b},\underline{c},j,k] \\ Candidate_2 &: [\underline{a},k,z,\underline{b},j,\underline{c}] \end{aligned}$$

the ROUGE-L score for C_1 and C_2will be the same but according to ROUGE-W C_1 is better because it contains a consecutive match of the LCS $[a,b,c]$. Two other metrics included in ROUGE are

ROUGE-S and ROUGE-SU. Both use skip-bigram overlap to calculate scores. A skip-bigram is a lexical feature similar to a bigram but it allows intermediate words between the words contained in a bigram. For example the following candidate sequence:

$$\begin{aligned} Reference &: [a,b,c,d] \\ Candidate &: [a,e,b,c] \end{aligned}$$

contains only one reference bigram $[b,c]$ but three skip-bigrams $\{[a,b],[a,c],[b,c]\}$. The ROUGE-SU extends ROUGE-S by introducing unigram overlap to differentiate between sentences that don't share any words at all and those that share only unigrams but not skip-bigrams.

2.9.3 Basic Elements

Another automatic evaluation that has been used during DUC 2006 and 2007 is the Basic Elements (BE) evaluation [50]. The main idea behind BE is to utilize parsers to enable a more intelligent overlap calculation. There exist several variations of BE available. The one used in DUC is BE-F, which utilizes Minipar [51] dependency parse trees. The word-relations in these trees are marked with labels such as subject, object, complement, modifier etc. Pairs of words with corresponding relations are then selected and used as basic elements for matching. BEs are weighted according to their appearance in the reference summaries. The matching process can be implemented at several levels such as exact match, match of root forms, WordNet similarity matching using synonymous relations, and match of semantic generalizations. In general, the approach used in BE evaluation is quite promising. However, in order to obtain results that highly correlate with manual methods like Pyramid Method, powerful matching routines and reasonably accurate BE weighting schemes are required.

2.9.4 Problems in Automatic Evaluation

Although automatic evaluation techniques such as ROUGE are widely-used in the research community, it turns out that the overlap even between manually constructed summaries can be very low, making ROUGE a less meaningful evaluation metric. Indeed, the problem of automatic evaluation of summaries is quite challenging and requires techniques that go beyond counting the superficial lexical features. In this respect a new task has been introduced in TAC 2009. The task, which is called Automatically Evaluating Summaries of Peers (AESOP), has the goal to promote the research on the development of systems for automatic evaluation of summaries. The results produced by the proposed evaluation techniques are compared against manual methods. In addition, the discriminative power of the metrics is tested. The idea is that a metric should correlate with manual

methods and reveal similar distinctions between summaries without transitive contradictions such as summary *a* is better than *b*, *b* is better than *c* but *c* is better than *a*.

3 Use of Graphs in Automatic Summarization

3.1 Graph Representations

A graph is a mathematical structure consisting of a set of vertices (also referred to as nodes) where vertices are connected by edges (also referred to as links, connections or relations). Formally a graph can be defined as follows:

$$G = (V, E) \tag{3.1}$$

where V is a set of vertices and E is a set of edges. Edges can be directed or undirected. An edge may have a weight associated with it. Graphs can be easily visualized. For example, consider an undirected graph in figure 3.1. Because the relations between nodes are explicitly represented by edges, both direct and latent relations can be easily identified.

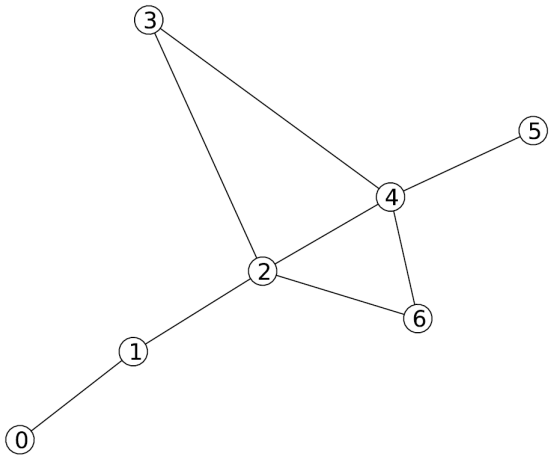

Figure 3.1: The Sample Graph.

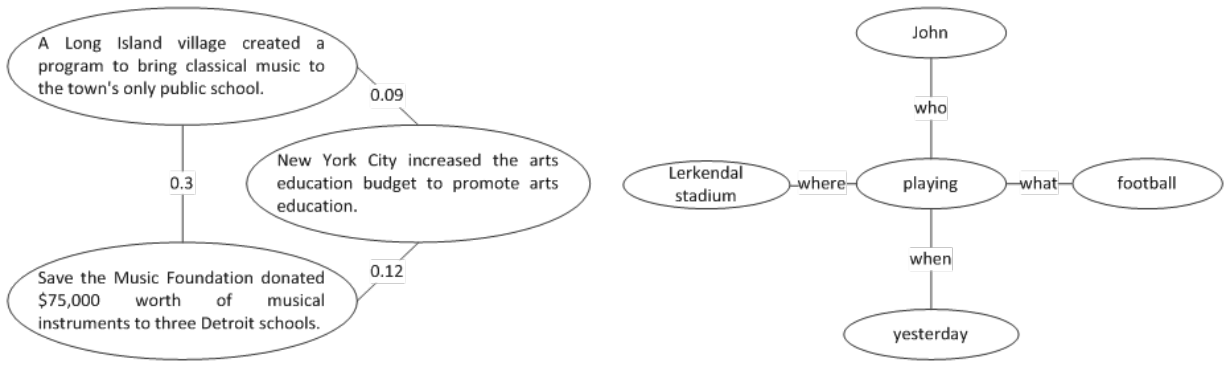

Figure 3.2: Sample similarity graph (left) and event graph (right).

Graphs are capable of representing a large variety of different phenomena such as hosts on the web, people connected by social links, pages connected by hyperlinks, settlements connected by transportation routes and so on. The real elegance of graph representations is that after a phenomenon is captured by a graph the generic techniques from the graph theory can be applied for its analysis. In general, graph representations are preferred in situations where the relations between objects are important. In case of automatic summarization a text is represented as a graph. There are different ways of representing a text as a graph. Two main questions that should be considered are:

- what is represented by vertices in a graph
- what kind of relations between vertices should be captured by edges

In order to decide upon these two issues one should think of the task at hand. The task of extraction-based summarization involves the creation of a summary by selecting textual contexts from the original documents. Then, the obvious choice for nodes are textual contexts such as paragraphs, sentences or words. With textual contexts as nodes the alternatives for relations represented by edges in a graph are:

- similarity relations
- semantic relations such as semantic roles, cause-consequences, specifications, time relations

Both of these alternatives are used in automatic summarization. Summarization approaches that make use of graphs with semantic relations are often referred to as event-based summarization. A sample graph that might be used in event-based summarizer is shown in figure 3.2 (left). In comparison with similarity graphs (see example in figure 3.2 right), event graphs capture semantic information in a more accurate way and in theory they are more useful. In practice, however,

accurate identification of semantic relations is a very complex task and therefore event-based summarization systems are quite rare. It is much easier to obtain similarity relations between contexts based on superficial lexical features. A variety of graph techniques can be applied to a similarity graph to exploit these relations. The next section elaborates on the centrality measures that can be used for content selection in extraction-based automatic summarization.

3.2 Centrality Measures

Graph theory and network analysis provide a great number of different methods and algorithms for working with graphs. For extraction-based summarization, methods for identification of the most salient sentences are required. When a text is represented as a similarity graph with sentences as nodes and similarities between sentences as edges, the problem of identifying salient sentences is transformed to the problem of identifying important or influential nodes in the graph. In graph theory the relative importance of a node in a graph is often referred to as node's centrality. There are several approaches for measuring the centrality of a node, and numerous variations of these approaches specially adapted for a specific task. Most of the basic measures and some advanced techniques are covered in the rest of this section.

To illustrate the difference between centrality measure, the sample graph shown in figure 3.1 will be used. Length of edges in this graph correspond to actual distances between nodes, which is inverse of similarity. The size of a node will be modified according to the centrality of this node calculated using a specific centrality measure, so that more central nodes will be larger.

3.2.1 Degree-based Methods

Degree is perhaps the simplest way to measure a node's importance in a graph. The degree of a node v is the number of edges attached to it:

$$deg(v) = \{\{v,u\} \in E : u \in V\} \tag{3.2}$$

The intuition behind this measure is that a node with many edges to other nodes should be considered more important than a node with few edges.

3.2.1.1 Degree Centrality

Although within one graph a degree by itself can be used to measure the relative importance of a node, when considering several graphs or subgraphs, a normalization factor is required. Degree

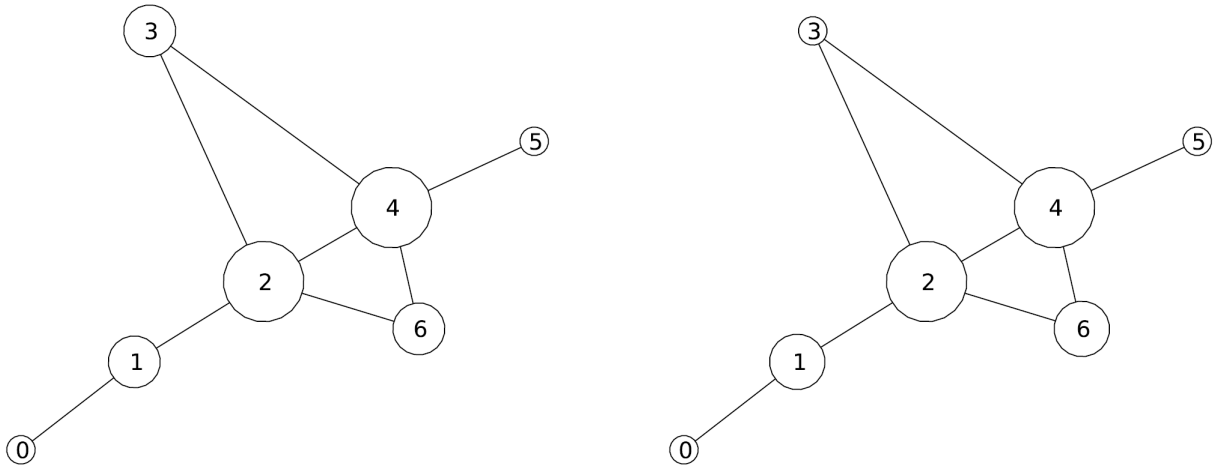

Figure 3.3: Degree (left) and weighted degree (right) centralities.

centrality of a node v is calculated by dividing the degree of a node $deg(v)$ by the number of nodes in a graph $|V|$ [52]:

$$C_d(v) = \frac{deg(v)}{|V|-1} \tag{3.3}$$

Despite its simplicity degree centrality is often a good indicator of a node's importance. In a graph with weighted edges, weights may indicate the importance of each connection. Thus the sum of the connection weights adjacent to a node v may be used as the centrality measure:

$$C_{wd}(v) = \sum_{e \in E(v)} weight(e) \tag{3.4}$$

where $E(v)$ is a set of edges directly connected to the node v. Further in this report C_{wd} will be referred to as weighted degree centrality.

Figure 3.3 illustrates the effect of the degree and the weighted degree centrality measures applied to the graph in figure 3.1. Degree centrality ignores the edge length and therefore nodes 3 and 6 with two edges each are of equal size. When calculating weighted degree centrality, edge lengths are taken into account, therefore, node 3, which is farther away from its neighbors, is smaller than all the other nodes.

3.2.1.2 Eigenvector Centrality

Eigenvector centrality [53] determines relative scores of all nodes in a graph where a connection to a high-scoring node contributes to the score more than a connection to a low-scoring node does.

The intuition behind the eigenvector centrality is that a node should be considered important if it has connections to other important nodes. It is quite different from the degree and weighted degree centralities, where centralities of adjacent nodes don't influence the centrality of the current node. By formulation, the centrality score of a node v is proportional to the scores of the nodes that are directly connected to node v:

$$C_e(v) = \frac{1}{\lambda} \sum_{u \in M(v)} C_e(u) \tag{3.5}$$

where λ is a constant, $M(v)$ is the set of nodes adjacent to v. The same equation in the matrix form, where the centrality scores are stored in a vector c and the graph is represented by an adjacency matrix A, can be written as an eigenvector equation:

$$\lambda c = A \cdot c \tag{3.6}$$

where scalar λ is an *eigenvalue* (corresponds to the previously mentioned constant λ) and c is an *eigenvector* (corresponds to centralities of nodes in a graph). Although eigenvector solutions exist for many different eigenvalues only the one where all the scores are positive is desired. Pursuant to *Perron–Frobenius theorem* this solution will correspond to the largest real eigenvalue. Finding the eigenvalues and eigenvectors of a matrix is an important problem in linear algebra and therefore many algorithms exist to deal with it. The *power iteration* algorithm and its modifications can effectively be used for this task.

Probably because of the efficiency of underlying idea and the flexibility of the approach, the eigenvector centrality and its modifications are used in a large variety of tasks where the identification of salient connected entities is required. Web page ranking is perhaps the most well-known application of eigenvector centrality. Although ranking of web pages is not the focus of our research this domain provides an excellent illustration of how the eigenvector centrality can be extended to accommodate a domain specific model.

Eigenvector centralities for nodes of the sample graph are shown in figure 3.4. Compared to the weighted degree centrality demonstrated in figure 3.3, nodes 5 and 0 have different sizes. Node 5 is larger because its only neighbor, node 4, is larger than the only neighbor of node 0, namely node 1.

3.2.1.3 PageRank

PageRank [54] is a variation of eigenvector centrality measure. Similar to eigenvector centrality it assigns scores to nodes in a graph, based on the scores of the connected nodes. However, the

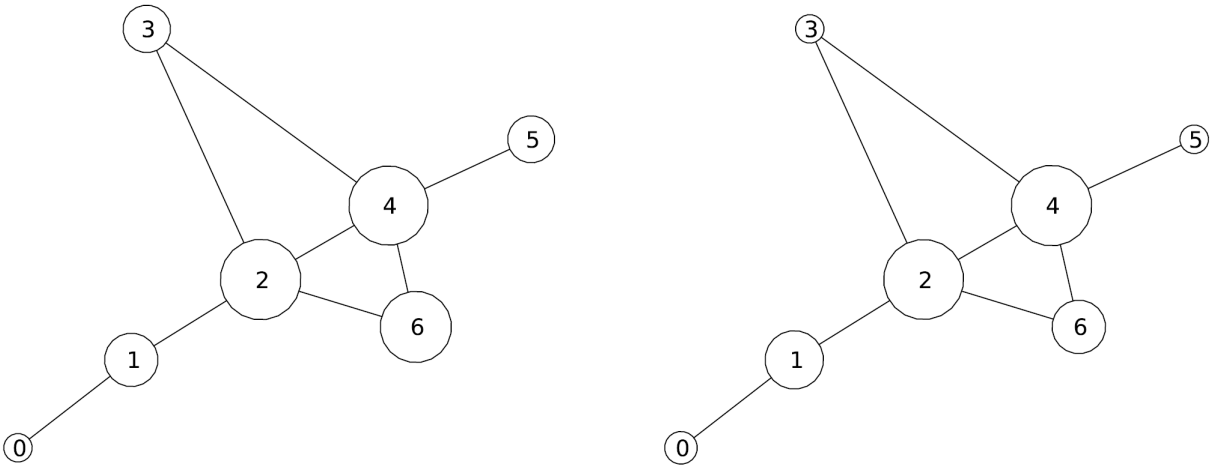

Figure 3.4: Eigenvector (left) and PageRank (right) centralities.

underlying model of PageRank is different and, in a sense, domain specific. The algorithm was proposed by Sergey Brin and Lawrence Page and is used by the Google search engine for weighting web pages based on the hyperlink connections with other web pages. In general, PageRank can be applied to any *directed graph* for assigning the relative centrality for every node in a graph. The algorithm employs the following formula for updating the score for a node v:

$$PR(v) = \frac{1-d}{|V|} + d \sum_{u \in V_{in}(v)} \frac{PR(u)}{|V_{out}(u)|} \tag{3.7}$$

where $|V|$ is the total number of nodes in a graph, $V_{in}(v)$ is the set of nodes that have an outgoing edge to node v (corresponds to an ingoing edge of node v), $|V_{out}(u)|$ is the number of outgoing edges of node u, d is a dumping factor. Although the equation is similar to eigenvector centrality equation, there are some new elements, such as the damping factor d, which is not present in the equation 3.5. To understand where these elements come from, it is necessary to consider the original application domain of PageRank, which is ranking of web pages connected by hyperlinks. The equation models, to some extend, the behavior of an agent surfing the web, where the PageRank weight $PR(v)$ represents the probability of the agent coming to page v. There are two ways for the agent to do so:

1. Use a hyperlink in another page u which leads to page v.
2. Stop using the hyperlinks and randomly switch to another page.

The damping factor d is the probability for the agent to use hyperlinks, and $1-d$ for random switching. Considering that when switching randomly the agent may end up in any web page in

the set V, the probability $1-d$ is divided by the number of web pages $|V|$. When using hyperlinks the probability of coming to page v equals the probability of being at page u divided by the number of links $|E_{out}(u)|$ in page u. This surfing model can be iteratively simulated to obtain the PageRank rankings. Alternatively, the model can be formulated as a eigenvector matrix equation similar to equation 3.6 and be solved by the power method. PageRank has been successfully applied to a variety of other domains such as determining the prestige of scientific journals [55], word sense disambiguation [56], automatic summarization [36] and many others. In some domains the use of directed links doesn't make much sense. Fortunately, PageRank proved to perform just fine with undirected graphs.

Figure 3.4 illustrates the results of applying PageRank to the graph in figure 3.1. Compared to eigenvector centrality, node 5 is smaller because of the denominator $|V_{out}(u)|$ in equation 3.7 that splits the influence of the central node 4 to several nodes.

3.2.1.4 HITS

Hyperlink-Induced Topic Search (HITS) [57] is another variation of eigenvector centrality measure originally designed for ranking of web pages. Although both PageRank and HITS operate on a set of pages connected by hyperlinks, there are some substantial differences between them. From the runtime perspective PageRank is meant to operate while indexing the web pages, reached by automatic indexing agent. HITS, on the contrary, is executed by a search engine as a response to a user. This distinction is important because HITS unlike PageRank does ranking of pages in the context of the query formulated by a user. HITS computes two different scores for each page: *hub score* and *authority score*. A page with a high hub score is supposed to have many links to pages with high authority scores while a page with a high authority score is pointed out by many pages with high hub scores. Similar to other eigenvector centrality approaches, HITS exhibits circular dependencies between scores where hubs and authorities provide mutual reinforcement to each other. On the level of mathematical abstraction, HITS operates on a *directed graph* where nodes with high hub scores have edges *to* many nodes with high authority scores and nodes with high authority scores have edges *from* nodes with high hub scores. Formally, these two scores for a node v are updated as follows:

$$authority(v) = \sum_{u \in V_{in}(v)} hub(u) \tag{3.8}$$

$$hub(v) = \sum_{u \in V_{out}(v)} authority(u) \tag{3.9}$$

where $V_{in}(v)$ is a set of nodes that have an edge to the node v, $V_{out}(v)$ is a set of nodes that have an edge from the node v. Starting with the hub and authority scores equal to 1, the algorithm updates

the scores iteratively. In order for the scores to converge they should be normalized at each iteration so that their square sums for all the nodes in a graph are equal to 1:

$$\sum_{v \in V} hub(v)^2 = 1 \tag{3.10}$$

$$\sum_{v \in V} authority(v)^2 = 1 \tag{3.11}$$

where V is the set of all nodes in the graph. In case of undirected graph $V_{out}(v)$ and $V_{in}(v)$ are identical and it doesn't make sense to calculate hub and authority scores separately, making HITS meaningless for undirected graphs.

3.2.1.5 GRASSHOPPER

GRASSHOPPER[38] (Graph Random-walk with Absorbing States that Hops among Peaks for Ranking) is a recent graph-based ranking algorithm. The major feature of this algorithm is the ability to encourage diversity among the top ranked nodes, that is, identifying nodes that are central but not strongly connected to other central nodes and share few neighbor nodes with them. It makes the algorithm especially effective when applied to a multi-document summarization where redundancy is a major issue.

The problem of joint optimization of relevance and redundancy was briefly discussed in section 2.7.3. GRASSHOPPER handles this problem gracefully by using absorbing Markov chains. The general idea of the algorithm is somewhat similar to PageRank where node centrality is defined by a stationary distribution of the random walk on a graph. The nodes with high visiting probabilities are considered to be the central ones. In GRASSHOPPER a similar Markov chain random walk is defined, but instead of using a stationary distribution as the final score for all the nodes, only the top node gets ranked. The ranked node is converted into an absorbing state, which makes it impossible to leave the node during a random walk. This way the visit probabilities of nodes that are tightly connected to the ranked nodes are decreased, allowing nodes in other parts of the graph to receive higher scores. The re-ranking procedure is repeated until the required number of the top ranked nodes is selected. The overall procedure is similar to MMR algorithm described in section 2.7.3 but executed directly on a similarity graph using a somewhat more consistent mathematical model.

Figure 3.5 illustrates the process of hopping from one centrality peak to another. After each jump, the previous peak gets reduced by turning the top node to the absorbing state. Formally, when node i is turned into the absorbing state the transitional probabilities change as follows: $P_{ii} = 1$,

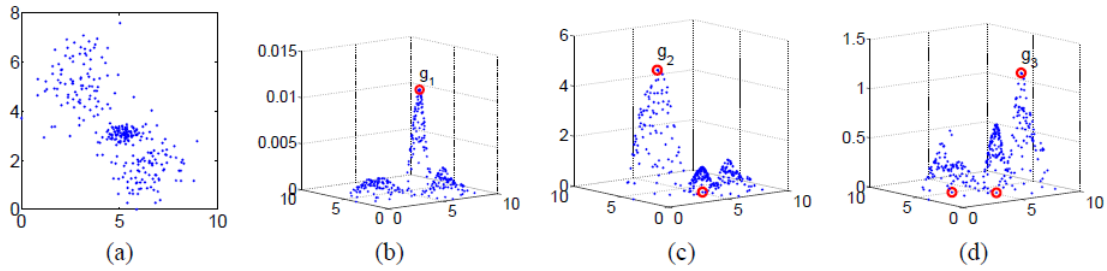

Figure 3.5: Sample data set (a) with stationary distributions before the first (b), second (c) and third iteration (d) (from the original paper [38]).

$P_{ij} = 0, \forall j \neq i$. Then the transitional matrix in canonical form can be arranged as shown below:

$$P = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix} \tag{3.12}$$

where the transitional probabilities for the ranked nodes are collected in the identity sub-matrix I (the transition is possible only to itself) and for the unranked nodes in sub-matrices Q and R. In order to calculate the number of visits to unranked nodes when the ranked nodes are represented by absorbing states the *fundamental matrix* is used:

$$N = (I - Q)^{-1} \tag{3.13}$$

where N_{ij} from N gives the expected number of visits to node i, if the random walk started in node j. The average over all possible starting nodes is used as the score in the current iteration. The score vector is calculated as follows:

$$\vec{S} = \frac{N^T \vec{1}}{n} \tag{3.14}$$

where $\vec{1}$ is the vector of 1 and n is the number of remaining unranked nodes. The experimental results demonstrate a good performance of GRASSHOPPER in different tasks including summarization.

3.2.2 Path-based Methods

There are several useful connectivity measures that utilize the concept of network path for assigning the centrality of a node. A path in a network is a sequence of nodes with edges connecting two neighbor nodes. Path based centrality methods often require finding the shortest path in a graph between two nodes. The shortest path, which sometimes is referred to as the geodesic path, is a path between two nodes containing a minimum number of nodes or edges among all the other

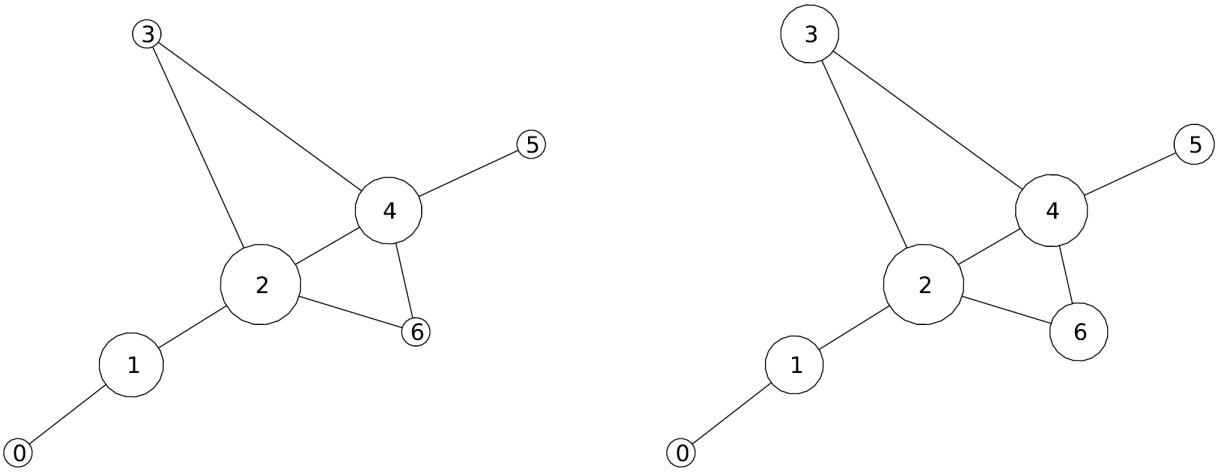

Figure 3.6: Betweenness (left) and closeness (right) centralities.

alternative paths between these nodes. In case of a weighted graph, a path with the minimum sum of weights is considered to be the shortest one. The problem of finding the shortest path is an important problem on its own. Several algorithms for solving the shortest path problem exist. Dijkstra's and Bellman–Ford algorithms are, perhaps, the most well-known ones. In this subsection two path-based centrality measures will be described and demonstrated on the Sample Graph.

3.2.2.1 Betweenness Centrality

Betweenness of a node is a fraction of the shortest paths between other nodes that goes through it. A node that occurs in many shortest paths is considered to be an important one. Formally, betweenness of a node v is calculated as follows [52]:

$$betweenness(v) = \sum_{u,t \in V : u \neq t \neq v} \frac{\sigma_{ut}(v)}{\sigma_{ut}} \tag{3.15}$$

where u and t are two other nodes, $\sigma_{ut}(v)$ is the number of shortest paths from u to t that goes through the node v, σ_{ut} is the total number of shortest paths from u to t. Betweenness may be normalized by dividing it by the number of node pairs not including the node v, which is $(|V|-1)(|V|-2)$ for the directed and $(|V|-1)(|V|-2)/2$ for the undirected graph where $|V|$ is the number of nodes in the graph. Betweenness centrality is calculated as follows (for directed graph):

$$C_b(v) = \frac{betweenness(v)}{(|V|-1)(|V|-2)} \tag{3.16}$$

Betweenness centralities of nodes in the Sample Graph are demonstrated in figure 3.6. Nodes 1, 2 and 4 are the most central ones because all the shortest paths goes through at least one of them. Removal of one of these nodes makes the graph disconnected.

3.2.2.2 Closeness Centrality

Closeness centrality [52] determines the centrality of a node based on a distance from the current node to all other nodes in a graph. A node which is relatively close to the other nodes is considered to be an important node. Closeness of a node v is calculated as the mean length of its shortest paths to all other nodes in the graph:

$$closeness(v) = \frac{\sum_{u \in V : u \neq v} d(v,u)}{|V| - 1} \tag{3.17}$$

where v is the current node, u is another node in the graph, $d(v,u)$ is the length of the shortest path between v and u, $|V|$ is the total number of nodes in the graph. The closeness centrality is the inverse of closeness:

$$C_c(v) = \frac{1}{closeness(v)} \tag{3.18}$$

When applied to the Sample Graph in figure 3.6, node 2 receives the highest score because it takes maximum 2 steps to reach any other node in the graph. Utmost nodes 0 and 5 are the least central ones according to closeness centrality.

3.2.3 K-Shell Decomposition

An interesting approach to finding the central nodes in a graph is provided by k-shell decomposition method [58]. The method consists of an algorithm that classifies nodes into k number of shells (layers), where shell values (layer depth) are used as the measure of centrality. The algorithm iteratively prunes nodes in a graph starting with the nodes with minimum degrees. First, all the nodes with the degree value 1 are pruned and assigned to $k_s = 1$ shell. This can cause other nodes with degree 1 to appear. They are also removed and assigned to $k_s = 1$ shell. When there is no nodes with degree 1 any more, the same process is repeated with degree 2, and the nodes are assigned to $k_s = 2$ shell. The degree is increased until all the nodes in the graph are removed. At the end, all nodes in the graph would be grouped in adjacent shells. The $k_s$value of a shell can be considered as the centrality value for all the nodes in this shell.

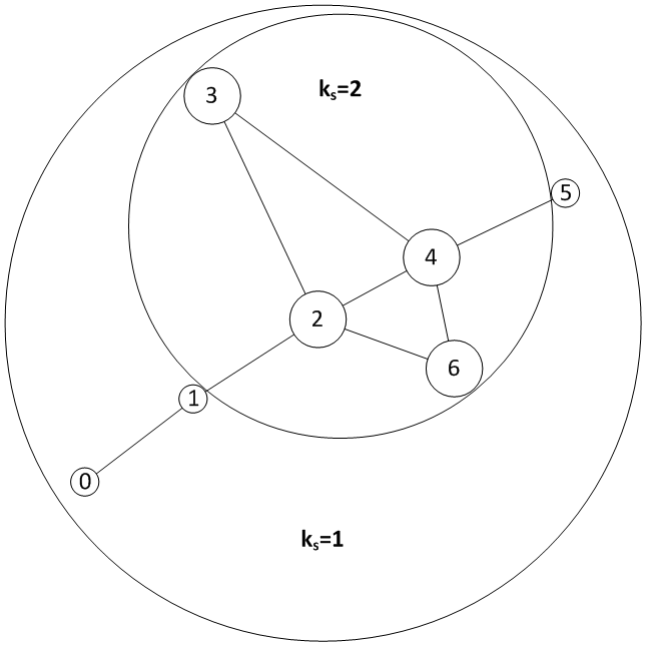

Figure 3.7: K-shell decomposition on a sample graph.

The Sample Graph after k-shell decomposition is shown in figure 3.7. First, the nodes 0 and 5, with degree 1 are pruned. After removal of node 0, node 1 will have degree equal to 1 and is also removed. No nodes with degree 1 are remained, so the first shell includes nodes 0, 1 and 5. Then, nodes 3 and 6 with degree 2 are removed. The remaining nodes 2 and 4 will then have the degree less than two and are also pruned. The second shell contains, hence, nodes 2, 3, 4 and 6.

3.2.4 Shapley Value for Top-k Nodes Problem

Shapley value is a concept from game theory that tightly connects to the notion of a coalition game. In a coalition game, several agents agree to cooperate with each other in order to obtain an overall gain in performance. The Shapley value provides a method for evaluating the contribution of each agent in a coalition. Suppose there is a set of agents N of cardinality n. The function $\upsilon(S)$ is defined for any subset S of N and provides the expected utility value attained by a coalition S without any help of other agents in N. The Shapley value calculates the contribution of an agent i by considering the increase in the utility value when the agent becomes a member of a coalition for all possible coalitions. Formally it is calculated as follows:

$$\Phi_i(\upsilon) = \frac{1}{n!} \sum_{\pi \in \Pi} [\upsilon(S_i(\pi) \cup i) - \upsilon(S_i(\pi))] \tag{3.19}$$

where Π is the set of all permutations over N, $S_i(\pi)$ is the set of agents preceding the ith agent in the permutation π. The sum is normalized by the number of possible coalitions $n!$.

The Shapley value is general enough to be used in a variety of tasks like feature selection for text classification [59] or top-k nodes problem in social networks [60]. The top-k nodes problem is related to local centrality measures and can be used to obtain a set of the most important nodes in a graph. Formally, the top-k nodes problem can be defined as follows:

$$S_{top} = \arg\max_{S \in \binom{V}{k}} g(S) \tag{3.20}$$

where V is a set of all nodes in the graph, $\binom{V}{k}$ is a subset of k distinct nodes in the graph, $g(S)$ is a function that provides the number of nodes adjacent to nodes in the subset S. Straightforward approach for finding top-k nodes is to try all the k-combinations of nodes in the graph and choose a subset S with the highest value of $g(S)$. Explicitly evaluating all k-combinations makes the problem NP-hard. Computing the exact Shapley value $\Phi_i(g)$ requires all possible $|V|!$ permutations of nodes to be considered, making the algorithm very inefficient. To avoid this, approximations of the Shapley value are used. Instead of considering all possible permutations over V, only a relatively small randomly sampled subset is used. After the Shapley values for all the nodes in a graph are computed, k nodes with the top values are selected.

Unlike other centrality measures discussed here, the formulation of the top-k nodes problem is directed towards finding the most influential group of nodes rather than individual nodes. This important distinction makes the top-k nodes approach suitable for the domains where coverage and diversity is an important issue. For example when applied to the extraction based multi-document summarization, the approach can potentially cope with the redundancy problems. For other domains such as ranking of retrieved web-pages this effect is undesirable.

4 The Extraction-based Summarization Framework

In the previous sections a variety of methods used in automatic summarization have been reviewed. Based on this acquired theoretical knowledge, we developed an extraction-based summarization framework (EBSF). The framework implements several approaches to summarization, and supports direct comparison of those. This section provides a description of the implemented framework.

4.1 System Overview

EBSF accommodates several approaches to extraction-based summarization, which can either be used separately or aggregated into one summarization workflow. Although designed for extraction-based query-focused multi-document summarization, the framework can be configured to perform other summarization tasks, such as single-document summarization or summarization without a query. The implemented summarizer is purely extractive, meaning that no sentence realization techniques mentioned in section 2.8 have been implemented. The reason for this is the complexity of such techniques, hence the time it would need to implement them. Sentence ordering is not implemented either.

The primary focus of the implemented system is, as with most other extraction-based summarization systems, the content selection mechanism, the procedure of deciding which sentences should be included in the summary. Typically, each sentence is assigned a score and sentences with high scores are selected. The overall workflow of the implemented system is shown in figure 4.1. Not all the steps that appear in the diagram are absolutely required to produce a summary. Some of the steps can be omitted depending on the configuration. The rest of this section describes each of the steps shown in the diagram.

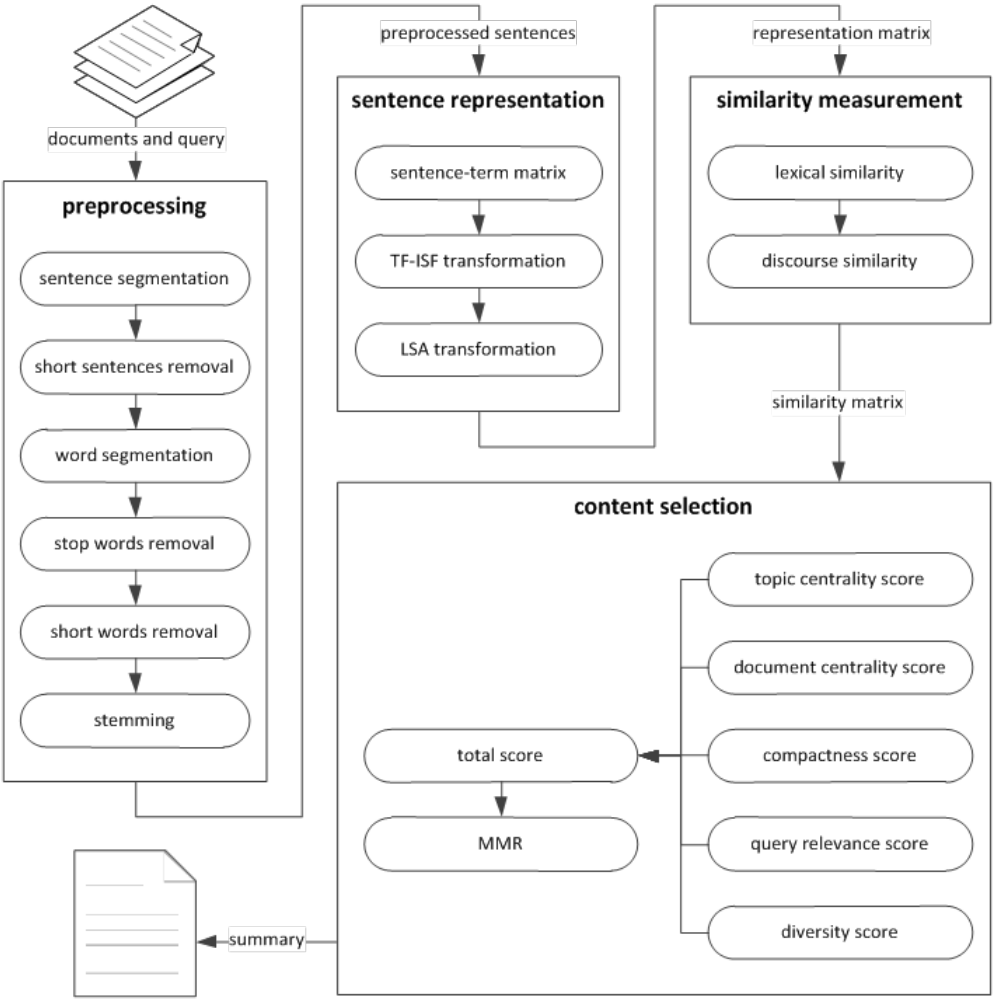

Figure 4.1: The workflow of the extraction-based summarization framework.

4.2 Preprocessing

The preprocessing stage involves most of the routines discussed in section 2.3. Routines utilized by the system are the following:

Sentence segmentation - splitting a text into sentences using unsupervised sentence boundary identification algorithm Punkt, described in section 2.3.4.

Short sentence removal - sentences containing less than three words are considered to be not informative enough or incorrectly segmented.

Word segmentation - splitting a sentence into words based on spaces and punctuation and using the conventions used by the Penn Treebank[1] to handle special cases, for example “weren’t”

[1]The Penn Treebank Project (http://www.cis.upenn.edu/~treebank/)

is split into two words “were” and “n’t”.

Stop word removal - the list of stop words used for the task is a combination of two lists: the one bundled with NLTK[2] and the one used by the ROUGE evaluation tool.

Short word removal - words smaller than three letters are considered to be non-content words.

Stemming - a rule-based stemming algorithm, the Porter stemmer, is used (described in section 2.3.1)

The input data for the preprocessing step is a collection of documents and a query. Preprocessing is applied to both of them. The output is a collection of sentences, where each sentence contains a set of processed words. The effect of the preprocessing can be seen on the following two sentences used as the input:

> Every weekend, students in Zhengzhou can take in a free concert, a traditional Chinese opera or a stage play in the city’s Youth and Children’s Palace, as part of a project to improve the younger generation’s education in the arts. The project was launched by the city government three months ago for students in local colleges, middle and primary schools.

The output produced by the system is the following:

> {weekend, student, zhengzhou, take, free, concert, tradit, chines, opera, stage, play, citi, youth, children, palac, part, project, improv, younger, gener, educ, art}
> {project, launch, citi, govern, three, month, ago, student, local, colleg, middl, primari, school}

From the sample output one can clearly see that most of non-content words are eliminated and the remaining words are reduced to their base forms. It directly effects further summarization steps by reducing the number of features used in the representation and providing generalization on the morphological level.

4.3 Sentence Representation

Sentence representation (context representation for sentences) and feature selection are tightly coupled and described in this subsection. Features are obtained by selecting unique words from the

[2]Natural Language Toolkit (http://www.nltk.org/)

preprocessed sentences. Each sentence is represented as a context vector. The vectors are accumulated into a representation matrix. The representation models implemented in the framework are the following:

Term count (TC) - represents sentences as vectors of which elements are the absolute frequency of words in the sentence.

Term frequency-inverse sentence frequency (TF-ISF) - the same weighting scheme as the TF-IDF described in section 2.4 but sentences are used instead of documents.

LSA - a distributed representation model described in section 2.4, the matrix V_k or the matrix product $S_k \cdot V_k$ obtained from SVD are used as the sentence representation matrix.

These representations can be used either separately or in combination. When combined, TC vectors are obtained first followed by TF-ISF and LSA transformations. The input to the sentence representation step is a set of preprocessed sentences and the output is the representation matrix where each row correspond to a particular sentence, columns are feature terms and cell values are weights. Sample output for TC model is provided below:

	art	concert	capit	citi	children	educ
s1	1	0	0	0	1	1
s2	1	1	1	0	0	0
s3	0	1	0	1	0	0
s4	0	0	0	1	0	1

The TC model often results in a very sparse matrix, because the number of term features is usually very large and sentences contain relatively few terms. Another observation is that the context vectors (rows) are nearly orthogonal and this leads to a very sparse similarity matrix.

The following TF-ISF representation is obtained from the matrix above:

	art	concert	capit	citi	children	educ
s1	0.23	0	0	0	0.46	0.23
s2	0.23	0.23	0.46	0	0	0
s3	0	0.35	0	0.35	0	0
s4	0	0	0	0.35	0	0.35

The sparseness of the representation remains the same but the values become weighted according to the TF-ISF weighting scheme. Maximum weights are assigned to terms that occur only in few

sentences and the lowest weights to terms that appear in many sentences. This scheme reflects the discriminating power of a term in a sentence. When LSA transformation is applied to the TF-ISF representation matrix with reduction to three dimensions, the output matrix takes the following form:

	d1	d2	d3
s1	-0.52	0.65	-0.48
s2	-0.52	-0.65	-0.48
s3	-0.48	-0.28	0.52
s4	-0.48	0.28	0.52

The obtained representation matrix is much less sparse than the other two above because the feature space is transformed to a concept space with smaller dimensionality. However, it is not a trivial task to determine the optimal number of dimensions analytically. Therefore it is done empirically by trying out several different values. When applied to a large number of sentences, LSA is able to detect latent relations between words and between sentences. The potential problem with LSA in summarization is that many sentences from the topically related documents are quite similar at the lexical level and the use of latent relations may result in even less precise similarity measurements between these sentences. Another interesting issue is the use of graph-based centrality measures together with LSA, when both are able to detect latent relations between sentences. Exploitation of the same latent relations twice may lead to undesired interferences between these two techniques.

4.4 Similarity Measurement

Similarity is computed between each pair of context vectors (rows) in the representation matrix. Similarities are used further in the summarization process to obtain centralities and to identify query relevant sentences. In the framework two different similarity metrics have been utilized:

Jaccard similarity coefficient - set-based similarity metric used for measuring similarities between sentences represented with TC representation.

Cosine similarity - a vector-based similarity metric used for representations with real-valued weights such as TF-ISF and LSA.

The output of this step is a similarity matrix of which both columns and rows correspond to sentences while the cell values indicate how similar these sentences are. The similarity matrix computed using the TC representation matrix and Jaccard coefficient is shown below:

	s1	s2	s3	s4
s1	1	0.2	0	0.25
s2	0.2	1	0.25	0
s3	0	0.25	1	0.33
s4	0.25	0	0.33	1

The matrix is symmetric with the diagonal values equal to 1.

In EBSF, we have extended the lexical based similarity with the notion of position. The extension exploits the intuition that sentences positioned near each other in a document are likely to express related information. In order to capture this intuition similarities between these sentences are artificially increased by multiplying the corresponding values in the similarity matrix. For example, assuming that sentences from the similarity matrix provided in this section are from the same document and appear in the same order as in the document, the similarity matrix can be modified in the following way:

	s1	s2	s3	s4
s1	1	0.4	0	0.25
s2	0.4	1	0.5	0
s3	0	0.5	1	0.67
s4	0.25	0	0.67	1

In this matrix, values for neighboring sentences are multiplied by two. The actual implementation is a bit more complicated than just simple multiplication. It allows to specify multiplication constant, radius of neighborhood and decay factor. This way sentences that are further away from each other get less increase in similarity than the direct neighbors.

4.5 Sentence Selection

Sentence selection is the final step in the construction of a summary in EBSF. The selection process mainly relies on similarities between sentences. Each sentence is assigned five different scores:

Query relevance score s_q is a similarity between a sentence and a query; sentences similar to the query get higher score. It represents how relevant is a particular sentence to the information requested by a query. If a query is not available this score is ignored.

Document centrality score s_{dc} measures the centrality of a sentence within the document where this sentence appears. First, a similarity graph is constructed from the similarity matrix for

the current document. Then a centrality algorithm is executed to obtain centralities for all nodes in this graph. Finally, the centralities are normalized. In single-document summarization this score alone can be used to identify sentences containing salient information. In multi-document summarization this score supports sentences that cover central topics of the documents they occur in. The intuition is that this score may help to select informative sentences from many documents rather than a few ones, thus providing a more comprehensive overview of topics covered in different documents.

Topic centrality score s_{cc} measures the centrality of a sentence within a collection of input documents. The computational procedure is almost the same as for the document centrality score described above. The only difference is that the similarity matrix used for the construction of the similarity graph contains similarities for all the sentences from all the documents in the collection. Assuming that the majority of documents in the collection are relevant to a specific topic, topic centrality score identifies sentences that contain the most salient part of the information shared by these documents.

Compactness score s_c is based on the length of a sentence; longer sentences get lower scores. The intuition is that a good summary should contain short but informative and relevant sentences. This score may be useful to choose between sentences of different sizes that express similar information.

Diversity score s_d is based on a similarity between a candidate sentence and sentences already included in the summary; less similar sentences get higher scores. This score is used by the MMR algorithm, described in section 2.7.3, in order to avoid redundancy.

The scores are aggregated into one final score that is calculated as the weighted sum of these scores:

$$s_f = w_q s_q + w_{dc} s_{dc} + w_{cc} s_{cc} + w_c s_c + w_d s_d \tag{4.1}$$

where w_q, w_{dc}, w_{cc}, w_c, w_d are weights for the query relevance, document centrality, topic centrality, compactness and diversity scores accordingly. Weights can be modified to vary the contribution of each score on the final score and thus the likelihood of a particular sentence to be included in a summary. The selection procedure follows MMR algorithm presented in section 2.7.3 where sentences with the highest scores are selected one at a time and the diversity score is recalculated for remaining candidate sentences. This way the redundancy is minimized in a greedy fashion. Centralities are computed using graph centrality algorithms, described in section 6, such as degree, weighted degree, eigenvector, PageRank, betweenness and closeness centrality. The output of this

step is a summary that contains sentences selected by the described procedure. The length of a summary is usually limited to a predefined number of words so the sentence selection procedure stops when this limit is reached.

5 Evaluation Method

This section describes the evaluation method used for the experiments presented in the next section. In general, our evaluation procedure conforms with the evaluation guidelines[1] from DUC 2007 competition. More details are provided below.

Evaluation of a summarization system is a non-trivial matter. A good way to do it is to participate in a TAC competition where manual methods reviewed in section 2.9.1 are utilized for evaluation. However, because of the time constraints, this was not a viable alternative for this thesis. Instead, automatic evaluation techniques were used. There are two things that should be considered in evaluation of a summarization system: input data and the evaluation method. The straightforward approach is to use datasets and evaluation tools from DUC and TAC competitions. The most recent competition that had used automatic evaluation is DUC 2007. The DUC 2007 dataset, as well as the summaries produced by the participants, is available upon request[2]. The main task[3] in the DUC 2007 competition is the query-focused multi-document summarization. The dataset for the task consists of 45 topics that contain a query and 25 documents each. The goal is to generate a query-focused summary for each topic. That is, a summary should provide the information requested in the query. Summary length is limited to 250 words. This task is similar to the real-life scenario when a collection of documents have been retrieved using document retrieval techniques and there is a need to summarize them according to the query.

DUC 2007 makes use of ROUGE evaluation package which is available upon request[4]. ROUGE calculates the n-gram overlap between a generated summary and reference summaries. Reference summaries were manually constructed by the experts for DUC 2007 competition. ROUGE includes several metrics based on different types of n-grams. Among them ROUGE-2 and ROUGE-SU4 are used for our evaluation procedure because they were also used in DUC 2007. Section 2.9.2 contains

[1] DUC 2007 guidelines are available at `http://www-nlpir.nist.gov/projects/duc/duc2007/tasks.html`

[2] DUC 2007 data is available at `http://www-nlpir.nist.gov/projects/duc/data.html`

[3] Full description of DUC 2007 tasks is available at `http://www-nlpir.nist.gov/projects/duc/duc2007/tasks.html`

[4] ROUGE evaluation package is available at `http://berouge.com/default.aspx`

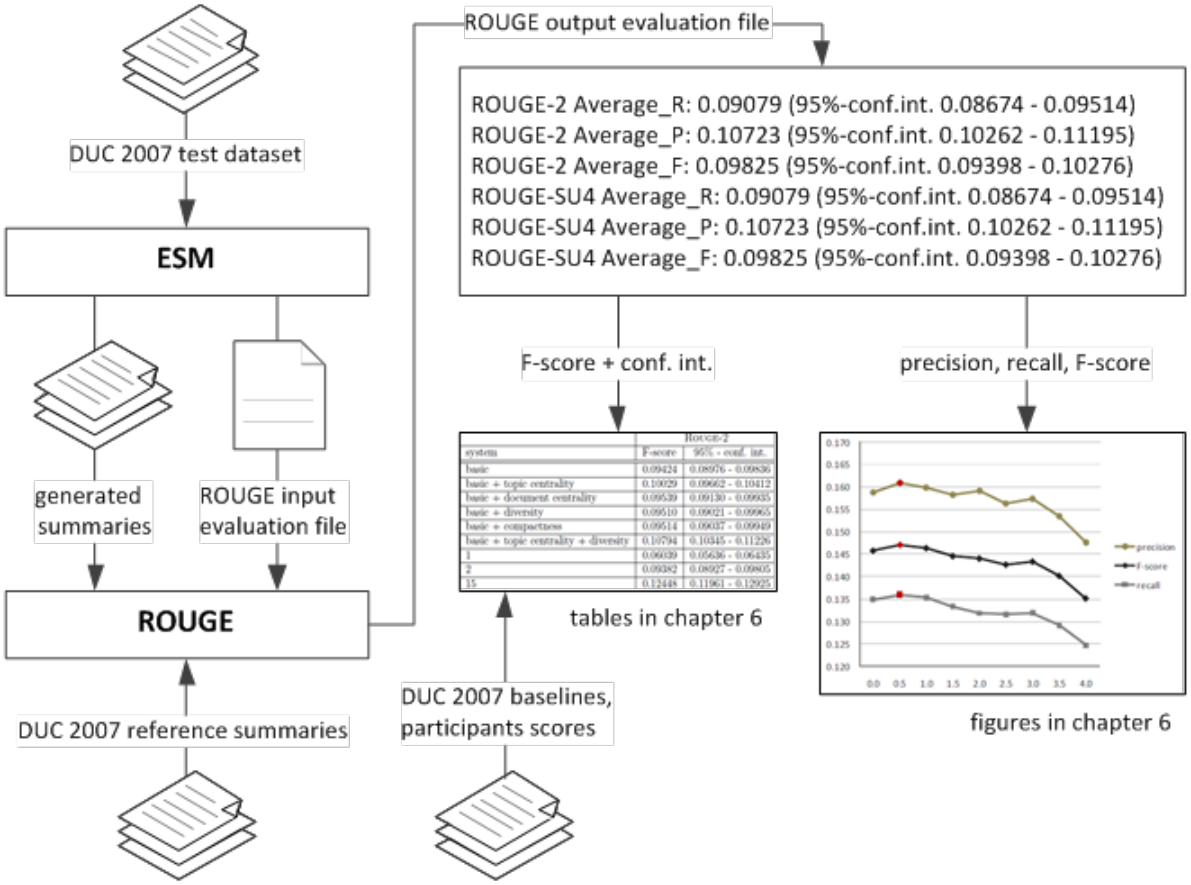

Figure 5.1: Evaluation workflow

the detailed description of these metrics. Use of DUC dataset and ROUGE evaluation allows the comparison of our system with baseline algorithms and summarization systems that participated in DUC 2007. Baselines used for evaluation are the following:

First DUC 2007 baseline retrieves the leading sentences from the most recent document. All the documents in the dataset have timestamps so identification of the most recent one is a trivial task.

Second DUC 2007 baseline is the automatic summarizer CLASSY04[61], which obtained the highest mean SSE[5] coverage score in DUC 2004 multi-document summarization task[6]. This summarizer ignores the query.

[5]SSE is a manual evaluation protocol. Additional information is available at `http://duc.nist.gov/duc2004/protocol.html`

[6]DUC 2004 task 2. Additional information is available at `http://duc.nist.gov/duc2004/tasks.html`

Basic Query-Focused Summarizer is our baseline summarizer, implemented using EBSF. The summarizer selects sentences that are most similar to the query. More details are provided in chapter 6.

The overall evaluation workflow for our experiments is shown in figure 5.1. First, the implemented extraction-based summarization framework (EBSF) is used to generate summaries for all the topics in DUC 2007 dataset. In addition, EBSF generates the evaluation input file for ROUGE. This file specifies which summaries should be evaluated and which reference summaries should be used for evaluation of those. Our input file implements jackknifing resampling method the same way as the official DUC 2007 input file. The idea of the method is to use several subsets of reference summaries to obtain a more robust statistical estimates of confidence intervals for means of the evaluation scores. Based on this file ROUGE script calculates precision, recall and F-score for each summary. Then the average scores with confidence intervals are computed. These scores were used to construct graphs and tables in section 6. For comparison, the results obtained by the baseline algorithms and some of the DUC 2007 participants are also presented in these tables.

6 Experiments, Results and Analysis

Although section 2 provides a general discussion of techniques that have been already investigated by many research groups in the field, there are very few papers that compare these techniques with each other. A typical summarization paper is usually dedicated to a single approach, which is described in detail and evaluated using automatic evaluation methods such as ROUGE or Basic Elements. The evaluation results provide an indication of a system's performance as a whole without deliberation on the reasons why a particular system performing better than another system. Therefore, research in this field is rather result oriented, leaving less space for comparative analysis of the techniques in controlled conditions. Automatic summarization is a complex, multistage process where several alternative techniques can be used at each stage. The goal of the experiments covered in this section is to investigate which of the techniques implemented in EBSF improve the quality of a summary and which don't. The reasonable way to do this is to try several alternative summarization workflows that use different techniques. The evaluation workflow described in section (5) provides quantitative values that reflect the quality of produced summaries. Comparison of these values as well as the manual inspection of the produced summaries constitute the bases for analyzing the effectiveness of the techniques with regard to the summarization task.

EBSF allows to modify a large number of configuration parameters that control the summarization workflow. The following parameters were the primary targets for the experiments:

Score weights control the contribution of the implemented scores on the sentence selection process. See section 4.5 for details.

Centrality measure specifies a centrality algorithm used for computing topic and document centrality scores. Discussion of centrality measures is provided in section 3.2

Representation model specifies a model for representing sentences. Representation models are discussed in section 6.

Other parameters are not as interesting to experiment with and are either assigned to some constant values or adjusted in accordance with parameters listed above.

	ROUGE-2		ROUGE-SU4	
summarizer	F-score	95% - conf. int.	F-score	95% - conf. int.
experiment 1	0.09424	0.08976 - 0.09836	0.14599	0.14155 - 0.15038
experiment 2.1	0.10029	0.09662 - 0.10412	0.15066	0.14652 - 0.15495
experiment 2.2	0.09539	0.09130 - 0.09935	0.14741	0.14330 - 0.15152
experiment 2.3	0.09510	0.09021 - 0.09965	0.14717	0.14245 - 0.15192
experiment 2.4	0.09514	0.09037 - 0.09949	0.14669	0.14216 - 0.15100
experiment 5	0.10794	0.10345 - 0.11226	0.15714	0.15264 - 0.16190
1	0.06039	0.05636 - 0.06435	0.10507	0.10076 - 0.10919
2	0.09382	0.08927 - 0.09805	0.14641	0.14224 - 0.15066
15	0.12448	0.11961 - 0.12925	0.17711	0.17244 - 0.18176

Table 6.1: Average F-score measured by ROUGE-2 and ROUGE-SU4 obtained by the basic query-oriented summarizer (BQFS), best preforming combinations of BQFS with four other scores (topic centrality, document centrality, diversity and compactness), DUC 2007 baseline systems (1 and 2) and the best performing DUC 2007 system (15).

Experiment 1: Basic Query-Focused Summarizer

In the first experiment a basic query-focused summarizer (BQFS) is tested. This summarizer selects sentences that are most similar to the query and, in a sense, is a traditional IR system with the difference that it deals with sentences rather than documents. BQFS is implemented using EBSF by setting all the score weights to 0 except the query relevance score weight, which is set to 1. BQFS uses the term count representation model to construct sentence vectors and Jaccard coefficient to measure how similar they are to the query. The performance of BQFS is shown in table 6.1. The results obtained by the summarizer are significantly (according to the 95% confidence interval) better than the first baseline (1) and (almost) the same as the second baseline (2), which is not bad considering that the second baseline is ranked 18 out of 32 DUC 2007 participants.

Experiment 2: Effect of Scores

In this series of experiments the effect of scores on the quality of generated summaries is investigated. Five scores are implemented in EBSF:

- query relevance score
- topic centrality score
- document centrality score

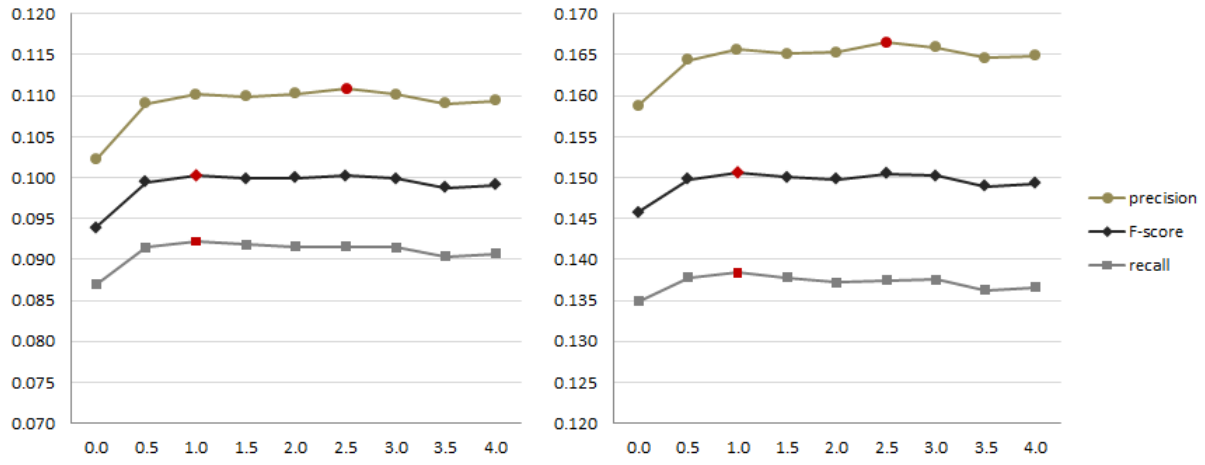

Figure 6.1: Results of experiment 2.1: average precision, recall and F-score measured by ROUGE-2(left) and ROUGE-SU4(right) depending on the topic centrality score weight (from 0.0 to 4.0).

- compactness score
- diversity score

These scores are used for ranking the sentences. Section 4.5 provides an overview of these scores and how they are combined together into the final score. Experiments described in this section have the goal to demonstrate the effect of each score on the quality of generated summaries. In order to do that, the corresponding weights, which determine how much a particular score contributes to the final score, are changed gradually and the generated summaries are evaluated.

The query relevance score is used in BQFS, which was already tested in the previous experiment. The hypothesis for this series of experiments was that when other scores are combined with the query relevance score the performance will be better than BQFS.

Experiment 2.1: Topic Centrality Score

Topic centrality is supposed to assign higher scores to sentences that are most similar to other sentences in a document collection. In the summarization task where no query is provided the topic centrality score can be used effectively to determine which sentences should be included in a summary. However, in a query-focused summarization the effect of this score is less obvious. In this experiment the topic centrality score is combined with the query relevance score. The centrality algorithm used in this experiment is the eigenvector centrality described in section 3.2.1.2.

The influence of the topic centrality score is gradually increased by modifying the corresponding score weight. The query relevance score weight is set to 1 and remains unchanged through all the

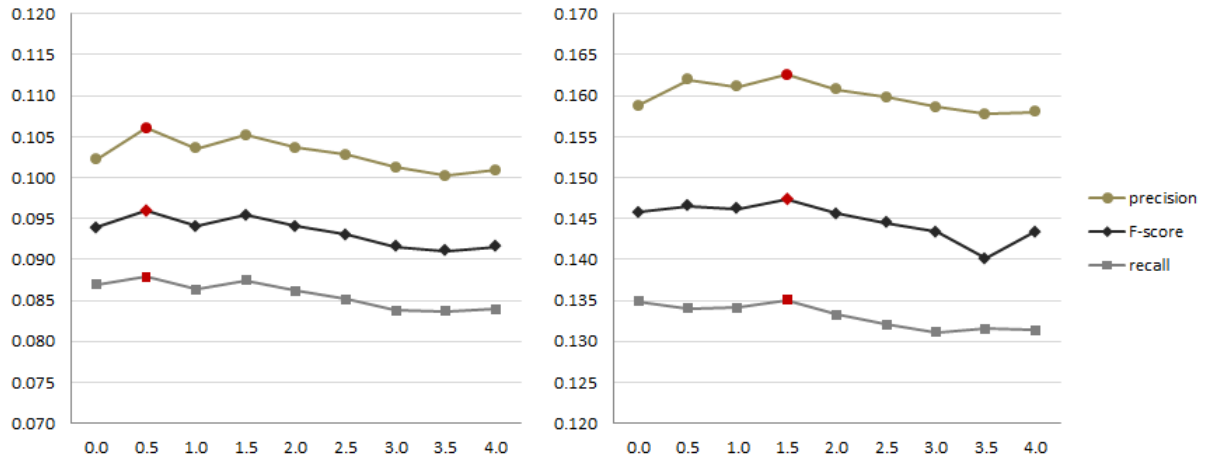

Figure 6.2: Results of experiment 2.2: average precision, recall and F-score measured by ROUGE-2(left) and ROUGE-SU4(right) depending on the document centrality score weight (from 0.0 to 4.0).

test runs. Figure 6.1 demonstrates the performance of the system relative to the topic centrality score weight. The best F-score is achieved when the weight equals 1, which is the same as the query relevance score weight. The exact F-score is shown in table 6.1, which is significantly better than both DUC 2007 baselines (1 and 2) and BQFS. Based on these result it is reasonable to conclude that query-focused summarization benefits from the introduction of the topic centrality component. According to optimal weights for this experiment the influence of topic centrality score on the final score should be the same as the influence of query relevance score.

Experiment 2.2: Document Centrality Score

Document centrality score assigns higher scores to sentences that are similar to other sentences in the same document. As opposed to the topic centrality, where salient sentences are identified relative to a collection of documents, the document centrality score assumes that when summarizing several documents each document contains important pieces of information that shouldn't be underrated when considering the whole collection. The centrality measure used in this experiment is eigenvector centrality and the overall setup is similar to the experiment 2.1.

The document centrality weight was gradually increased in order to investigate the influence of document centrality score on the overall performance. The results obtained from this experiment are presented in figure 6.2. It can be seen that the best F-score is achieved when the corresponding weight takes the value 1.5[1], which, according to table 6.1 is higher than for both baselines (1 and

[1]According to ROUGE-2 value 0.5 gives higher F-score but when ROUGE-2 and ROUGE-SU4 are combined

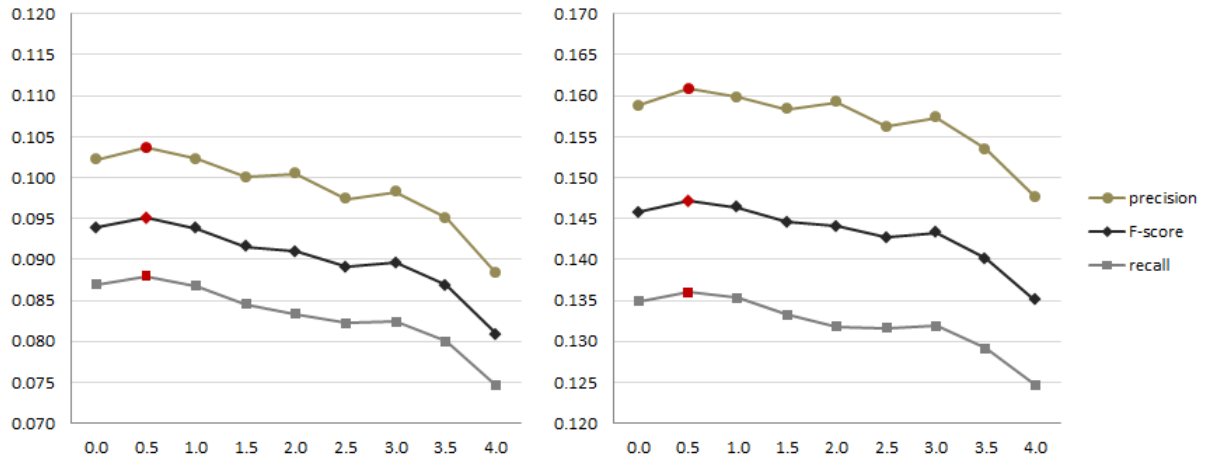

Figure 6.3: Results of experiment 2.3: average precision, recall and F-score measured by ROUGE-2(left) and ROUGE-SU4(right) depending on diversity score weight (from 0.0 to 4.0).

2) and BQFS. However, the difference is not significant according to the 95% confidence intervals shown in table 6.1. In general, the conclusion for this experiment is that the document centrality score might be able to improve the performance of a query-focused summarizer when the influence of the score is higher than that of the query relevance score. It increases the chance that the relevant information will be selected from several documents, which may result in an incoherent summary but at the same time making the summary, in a sense, more comprehensive.

Experiment 2.3: Diversity Score

The diversity score is used to minimize the redundancy in a summary. It is a part of the MMR algorithm described in section 2.7.3, where sentences that are similar to the sentences that have already been included in the summary have a lower chance to be selected. The experiment investigates the effect of the diversity score on the overall performance of the system. The setup is the same as for BQFS except that the diversity score weight is larger than zero and is gradually increasing in each test run. This way the optimal value is determined.

According to the results presented in figure 6.3 the maximum performance is reached when the influence of the query score (default weight 1.0) is twice as much as of the diversity score (weight 0.5). The exact F-score and 95% confidence interval are shown in table 6.1. The performance is slightly better than the performance of BQFS and of the second baseline (2) and significantly better than the first baseline (1).

F-score for value 1.5 is higher.

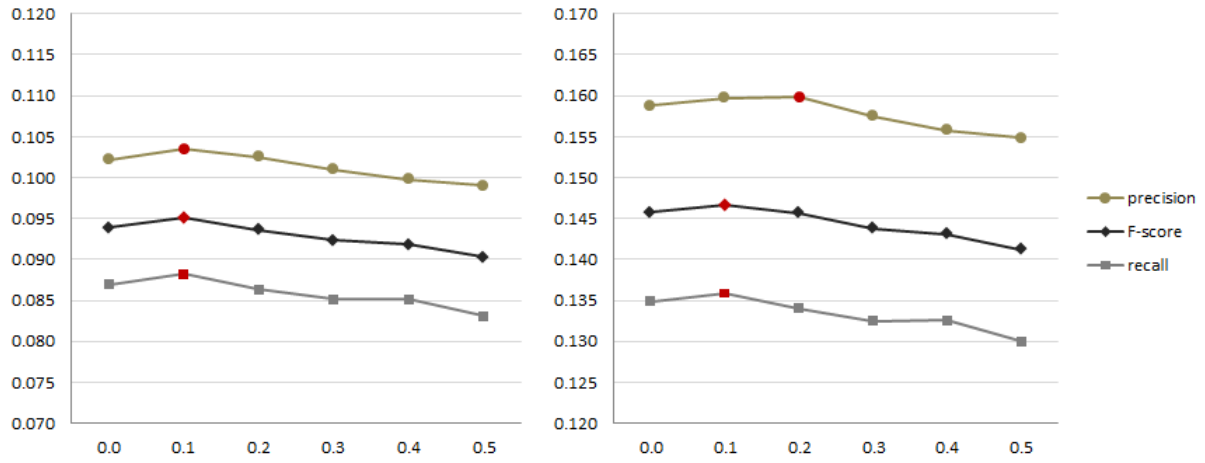

Figure 6.4: Results of experiment 2.4: average precision, recall and F-score measured by ROUGE-2(left) and ROUGE-SU4(right) depending on the sentence compactness score weight (from 0.0 to 0.5).

The diversity score is part of the MMR procedure targeting to avoid the redundancy. It was expected that the use of this algorithm would lead to a substantial increase in the recall, however, in our experiments both recall and precision increase only a bit. Although the results show merely a rather modest increase in performance, optimization of diversity is an essential component in multi-document summarization. Perhaps, in more realistic conditions where a summary length is not limited to only 250 words it might render to be more useful.

Experiment 2.4: Compactness Score

The sentence compactness score has a simple utility of assigning higher scores to shorter sentences. On it's own it is useless but combined with other scores it introduces a bias towards shorter sentences. For the summarization task where the length of a summary is limited, one can expect to see its benefits. The setup for the experiment is exactly the same as for BQFS (6), but the weight of the sentence compactness score is larger than zero. As with other experiments, this weight was increased gradually to observe the effect it makes on the overall performance.

Figure 6.4 demonstrates the results of the experiment. Maximum performance is achieved when the weight has the value 0.5, that is half of the query relevance score weight. The exact F-score provided in table 6.1 is insignificantly better than the performance of BQFS and the second baseline. Our assumption is that the score will be more useful for larger document collections where several candidate sentences with approximately more or less the same information but with different grammatical structures and sizes are presented to the content selection component. The

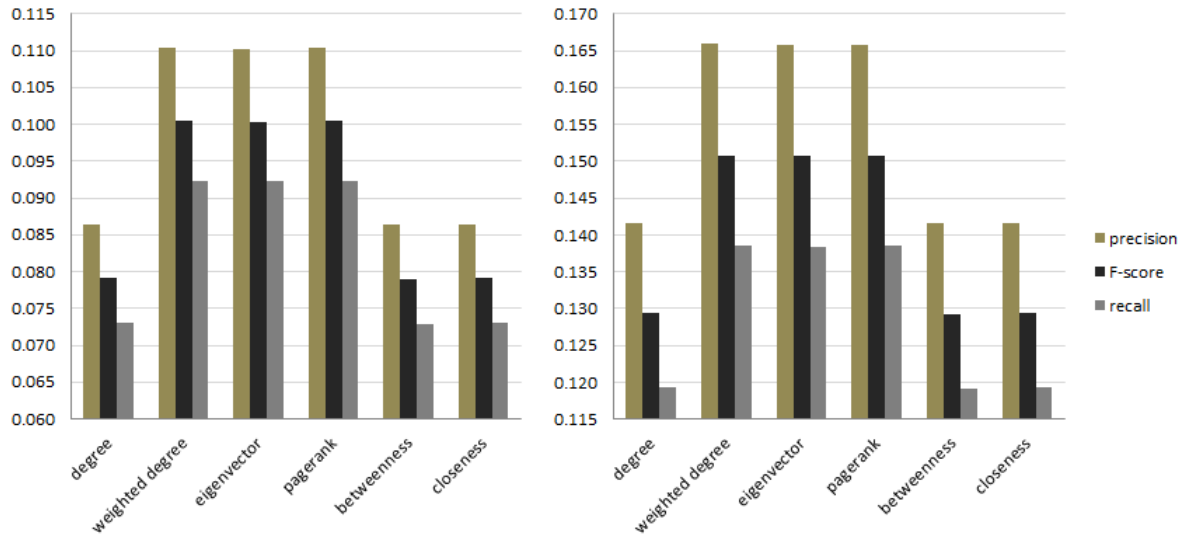

Figure 6.5: Results of experiment 3: average precision, recall and F-score measured by ROUGE-2(left) and ROUGE-SU4(right) achieved with different centrality measures.

same situation is likely to occur when a sentence simplification algorithm is applied, when several alternative simplifications as well as the original sentence compete with each other. In this case a relevant but concise sentence might be selected based on the query relevance and sentence compactness scores.

Experiment 3: Centrality Measures

Section 3.2 described a variety of centrality measures. In the experiments with topic and document centrality scores it was observed that the use of centralities improve the performance of the query-focused summarizer. A significant improvement was observed with the topic centrality score weight set to 1, same as the query relevance score weight. A similar setup is also used in the current experiment, but unlike my previous experiments, where only the eigenvector centrality is used, in the current experiment several well-known centrality measures are tested. The goal is to compare their performance in the query-focused summarization task.

Six different centralities have been evaluated in this experiment. Four of them are degree-based centralities (degree, weighted degree, eigenvector, PageRank) and two are path based centralities (betweenness, closeness). The results obtained by the summarizer are shown in figure 6.5. Path-based centralities performed significantly worse than degree-based ones except degree centrality itself. Possible reason for this may be that path-based centralities exploit latent(indirect) similarity relations between sentences, which can be quite distracting when the similarities are not strong

enough. Perhaps, use of a similarity threshold, discarding weak relations between sentences is a reasonable remedy. For the same reason the degree centrality demonstrated the performance as weak as path-based centralities. Without the appropriate threshold the number of sentences that have at least one content word in common is not a good indicator of a sentence salience. Other degree-based centralities (weighted degree, eigenvector, PageRank) make use of relation strengths and are less distracted by weak relations. The performance of these three centrality measures is almost the same, although there is a substantial difference in the mechanism of weighted degree centrality and the other two. Eigenvector and PageRank make use of indirect relations, while weighted degree centrality does not. Although the difference in their performance is very minor they can be ranked as follows:

1. Weighted degree centrality (ROUGE-2 F-score 0.10046 , ROUGE-SU4 F-score 0.15082)
2. PageRank centrality (ROUGE-2 F-score 0.10045, ROUGE-SU4 F-score 0.15079)
3. Eigenvector centrality (ROUGE-2 F-score 0.10029, ROUGE-SU4 F-score 0.15066)

The fact that weighted degree centrality performs at least not worse (slightly better) than PageRank and eigenvector confirms the hypothesis that the latent relations are not that useful and might be distracting in query-focused summarization.

It is also worth mentioning that in terms of algorithmic complexity weighted degree is much simpler than all the other centrality measures except the degree centrality. Calculation of shortest paths in closeness and betweenness measures is an expensive operation on a large graph. Power iteration methods used for finding eigenvectors in PageRank and the eigenvector centrality require less computational resources but are still far from the simplicity of weighted degree centrality which just takes the sum of weights of direct connections.

Experiment 4: Representation Models

Another important aspect in summarization and in NLP are text representation models used in the construction of context/sentence vectors. Theoretical discussion of this aspect is provided in section 2.5. In this experiment the performance of several representation models is investigated with regard to the query-focused summarization task. Models tested in the experiment are:

- term count vector space model (TC)
- latent semantic analysis, reduced to 10 dimensions (LSA 10)

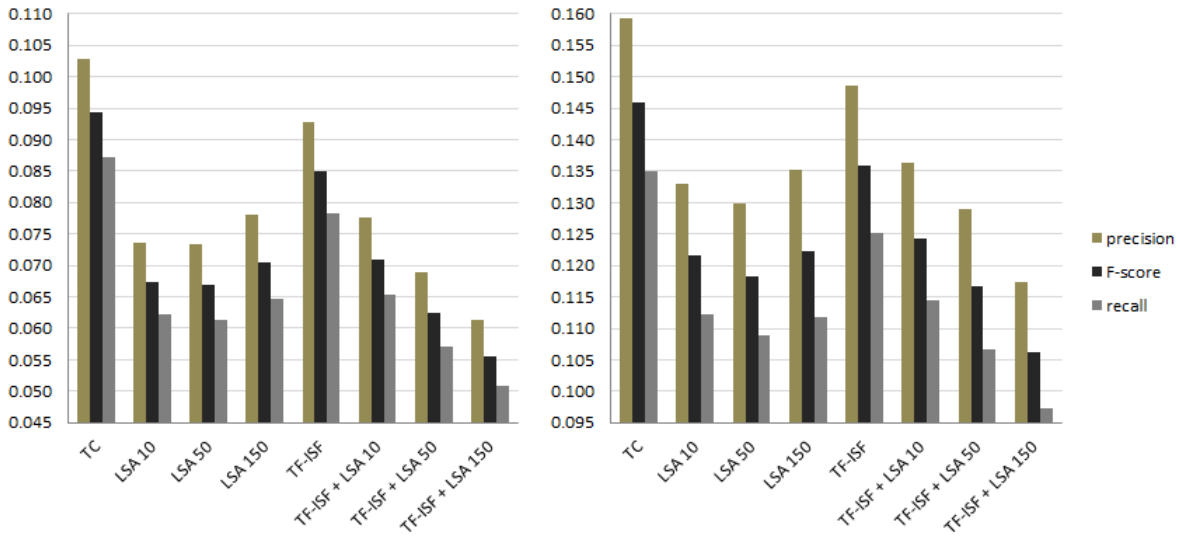

Figure 6.6: Results of experiment 4: Average precision, recall and F-score measured by ROUGE-2(left) and ROUGE-SU4(right) achieved with different representation models.

- LSA, reduced to 50 dimensions (LSA 50)
- LSA, reduced to 150 dimensions (LSA 150)
- term frequency - inverse sentence frequency (TF-ISF)
- TF-ISF combined with LSA, reduced to 10 dimensions (TF-ISF + LSA 10)
- TF-ISF combined with LSA, reduced to 50 dimensions (TF-ISF + LSA 50)
- TF-ISF combined with LSA, reduced to 150 dimensions (TF-ISF + LSA 150)

The setup used for the experiment is similar to BQFS, with the exception of the utilized representation models, which are different in each test run. Test runs with the TC model make use of Jaccard coefficient as the similarity metric while all other models use cosine similarity. This is a reasonable decision because vectors in TC model are mostly boolean vectors (rarely the same content word occurs more than ones in a sentence) and Jaccard performs well in this conditions, while cosine is a standard similarity metric in NLP when dealing with real valued context vectors.

The results obtained from the test runs are presented in figure 6.6. Even without the detailed analysis of the chart it is clear that LSA and the combination of LSA with TF-ISF are unable to deliver adequate results in BQFS approach. This is quite surprising considering the success of LSA in a large spectrum of NLP tasks. In order to clarify the situation, several summaries and the evaluation

results obtained for these summaries were inspected manually. The inspection revealed that summaries generated using LSA models may contain sentences that are relevant to the general topic of a query but may not necessarily contain the terms from the query. This observation aligns well with the mechanism underlying LSA; it reveals latent relations between terms through collocations with other terms. Considering that all the documents provided as input to the summarizer are at least partly relevant to the query, the use of latent relations may decrease the precision of the content selection component. The situation is similar to the one described in the analysis of centrality measures in experiment 3. Although it sounds like a consistent explanation of poor LSA performance it doesn't mean that automatic summarization can't benefit from LSA. Previously it has been shown that LSA [20] can be successfully used in multi-document summarization but with a significantly different approach and in a slightly different summarization task. DUC 2007 dataset, which is used for our experiments, includes long, detailed queries (or topic definitions in terms of DUC), so the relevant sentences tend to have a relatively large lexical overlap with the query and therefore no exploitation of latent relations is required. For more concise queries LSA will probably be more useful. The summary length limitation of 250 words can also be an important factor here because it is not large enough to accommodate relevant but indirectly related sentences.

The TF-ISF weighting scheme also performed significantly worse than the primitive term count vector space representation. TF-ISF is the adaptation of the TF-IDF scheme that has long been successfully used in document-oriented tasks but it turns out that it is not appropriate when dealing with sentences. The term frequency part of the weight introduces the length of a sentence into equation 2.2, so that words in longer sentences have less weight than in a shorter ones. Although some bias towards shorter sentences may be useful it should be carefully controlled, like when using compactness score, because short sentences tend to be highly dependent on the information contained in other sentences that might not been included in a summary. The inverse sentence frequency part is supposed to assign higher weights to terms rarely seen in other sentences, which can be a problem considering that all the documents in a collection are topic related. This way the important terms tend to occur frequently and will be underrated. In general, when dealing with small samples of data like sentences, statistically weak schemes such as TF-ISF may provide poor results. A reasonable alternative is to use weighting schemes based on the likelihood ratio described in section 2.4.

Experiment 5: Optimizing Performance

In this last experiment the acquired theoretical and empirical knowledge are utilized to achieve better performance by combining some of the techniques that demonstrated their potential in the

previous experiments. Among all the tested extensions of BQFS only the topic centrality score improves the performance significantly. The maximum increase in the performance is observed when the weight of the query relevance score is equal to the weight of the topic centrality score. We will exploit this finding by setting both weights to 1 in this experiment.

The document centrality score could be useful for practical reasons discussed in section 6 but it doesn't improve the performance significantly. In several test runs when combined with the topic centrality score in addition to the query relevance score, it actually worsened the evaluation results. Therefore, it is not included in the setup for the current experiment. Both compactness and diversity scores have minor influence on the quality of the generated summaries, but from the theoretical perspective it is obvious that multi-document summarization requires a mechanism to avoid redundancy. Based on this assumption the diversity score weight should be set to a non zero value. In the experiment with the diversity score the maximum performance was achieved when the diversity score weight is set to half of the query relevance score weight. Considering that in this experiment the topic centrality score is used in addition to the query relevance score, the weight for the diversity score should be at least doubled. Several test runs revealed that the best performance is achieved when the diversity score weight is a bit over 1, which aligns well with our analysis. The weights used for the setup in this experiment have the following values:

- query relevance score weight = 1
- topic centrality score weight = 1
- diversity score weight = 1.2
- document centrality score weight = 0
- compactness score weight = 0

Among the centrality measures tested in experiment 3, weighted degree, PageRank and eigenvector performed much better than the other ones. Top performance of weighted degree centrality combined with its low computational complexity makes it a perfect candidate for the current experiment. Among the representation models tested in experiment 4, the term count vector space model with Jaccard similarity metric yielded better results than the other ones and therefore is used in this experiment. The rest of the parameters are the same as in BQFS.

The evaluation results for the system are shown in table 6.1 (experiment 5). The results are significantly better than both baselines and the previously tested combination of the query relevance score with the topic centrality score (experiment 2.1). The comparison with the results from DUC 2007 participants is summarized in table 6.2. As we can see from the table, the setup used in this

	ROUGE-2	ROUGE-SU4
better than	24	20
worse than	8	12
significantly better than	20	17
significantly worse than	4	7

Table 6.2: Comparison with DUC 2007 participating systems.

experiment is quite promising. It performs significantly better than the majority of the DUC 2007 systems, especially according to ROUGE-2 evaluation metric, where it is significantly better than 20 of 32 systems.

6.1 Summary Quality

In this subsection some quality aspects of the generated summaries will be discussed. To be more concrete one of the summaries generated in experiment 5 is presented here:

Query Describe the state of teaching art and music in public schools around the world. Indicate problems, progress and failures.

Summary Dance is an indispensable part of the curriculum for all 750 students at Public School 156, along with music, art and creative writing. Linkup was developed in the 1980s when arts programs were being slashed in New York City public schools and elsewhere around the country. Delaine Easton, the State Superintendent of Public Instruction, has called for the restoration of arts education in California public schools. The study also found that as students progress through high school they are less likely to be involved in the arts. For the last two decades, schools like Public School 156 have increasingly turned to the city's museums, concert halls and theaters for lessons in art, music and dance. The district provides transportation and security and has helped the school create a program for teaching English through art. On the national level, Florida toddlers now listen to classical music in public schools by law. Steel drum has gained momentum in the city's public schools, which received $25 million last September from the administration of Mayor Rudolph Giuliani to restore arts and music programs that were cut several years ago. Under the new requirements, incoming CSU freshmen will be required to have an extra year of high school laboratory science and another of social science; UC will require a year of art, dance or music classes.

The generated summary contains a collection of facts relevant to the query and, in general, quite useful to draw query relevant generalizations. Ideally these generalizations should be extracted from the source documents by selecting the appropriate sentences but it is, perhaps, too much to expect from an extraction-based summarizer. The generated extract provides only the supporting facts and generalizations should be done manually based on these facts. Each sentence represents a separate fact and there is no coherence relations between sentences. Perhaps some form of content ordering based on similarities between sentences included in the summary might be utilized to improve the coherence.

Several sentences contain unresolved references that may cause difficulties in understanding. For example:

> Under the new requirements, incoming CSU freshmen will be required to have an extra year of high school laboratory science and another of social science; UC will require a year of art, dance or music classes.

It is not clear what requirements this sentence is referring to because previous sentences that specify these requirements are not included in the summary. The detection of this reference is a complicated task and requires advanced reference detection techniques. The most straightforward approach for resolving this reference is to include previous sentences that contain information about these "new requirements". However, given the limited length of the summary avoiding sentences with references may be a better decision.

Another problem of the generated summary is the lack of important details from the global context (document) of each sentence. In the query it is explicitly specified that the information about art and music in public schools may come from different parts of the world. In this regard, sentences may be augmented with the places where the given information applies. The place information can be extracted from the document where the sentence comes from.

In general, most of the generated summaries have the outlined problems and a lot can be done to improve their quality. The inspection of summaries generated by top DUC 2007 systems reveal the fact that they make use of sentence simplification techniques, which are known to improve ROUGE scores but may result in grammatically incorrect sentences.

7 Conclusion

This thesis provides a comprehensive review of techniques used for the task of extraction-based automatic summarization. As it becomes clear from the theoretical part of the thesis, automatic summarization is a complex task that involves several challenging subtasks. The performance in each of these subtasks directly affects the ability to generate high quality summaries. In extraction-based summarization the key part of the process is the identification of salient units of text. Graph-based representations and centrality methods are known to perform well in resolving the general problem of identifying important entities based on the relations between these entities. When applied to the content selection subtask, centrality measures are used to exploit similarity relations between sentences to select the most central ones. The underlying mechanisms of several different centrality measures were thoroughly discussed in this report.

In the practical part of this thesis, the EBSF framework was designed and implemented. The framework implements several approaches to extraction-based summarization and can generate summaries for various summarization tasks including query-focused multi-document summarization. Using DUC 2007 data a series of experiments was conducted with a variety of techniques applied in each of them. Among the techniques tested in the experiments are text representation models, centrality measures, size optimization and the redundancy removal approach. The dedicated experiments with each of these techniques served as the basis for analyzing their effectiveness with regard to the automatic summarization task. After the influence of these techniques on the quality of the generated summaries became clearer, several of them were combined in order to optimize the performance of the summarizer. The obtained results were compared to the results of DUC 2007 participants. The comparison demonstrated that the implemented system can deliver quite good results in the query-focused multi-document summarization task.

Some of the interesting conclusions of the theoretical and empirical studies are:

1. Centrality measures combined with query similarity can be successfully used for query-focused extraction-based summarization.

2. Further improvement of summarization performance requires the use of sentence realization techniques such as sentence simplification.

3. Among different centrality measures, the degree-based ones perform substantially better for content selection compared to path-based centralities. Although eigenvector and PageRank centralities utilize more powerful algorithms, their performance is not better than that of the weighted degree centrality, which is a much simpler centrality measure.

4. The straightforward use of LSA representation in query-focused summarization provides unsatisfactory results. Simple term count vector space representation model with Jaccard similarity metric performs better than the TF-ISF weighting scheme and LSA with cosine similarity.

5. Items 2 and 3 lead to the conclusion that the effective exploitation of latent similarity relations between sentences for generation of query-focused summaries is probably a more sophisticated task than originally assumed. In the implemented system the use of these relations lead to summaries that were less query-focused and therefore less precise.

7.1 Future Work

Automatic summarization is an active area of research with many aspects that can be investigated. With regard to the work done in this thesis the following issues can be considered to be the primary ones in future work:

1. Considering the success of LSA in other NLP tasks, it is certainly worth doing a further research on the effective use of LSA in automatic summarization. One of the conclusions of the thesis is that the straightforward approach of using LSA doesn't work well in a query-focused extractive summarization. With LSA, summaries become less query-focused, which in theory could be avoided by prioritizing dimensions directly relevant to the query. The alternative is to combine LSA with local representations. Besides LSA, there is another promising distributed representation discussed in the theoretical part of this thesis, which is random indexing. It would be interesting to include RI in the representation model experiments, and see how it performs in comparison with other representations.

2. Feature selection is another important aspect of automatic summarization that can be addressed in more detail. In the implemented system only unigrams are used as features despite the fact that ROUGE evaluation methods make use of bigrams and skip-bigrams as well as

unigrams. Obviously, the introduction of this features can potentially improve performance of the systems when evaluated with ROUGE. Theoretical discussion of how the bigrams can be extracted using log-likelihood statistics is provided but not implemented and tested in the system. Log-likelihood weighting scheme can be also used for unigrams instead of TF-ISF scheme, which have demonstrated unsatisfactory performance in the experiments.

3. Several major summarization subtasks such as sentence ordering and sentence realization were not implemented in the system. These are very interesting and constitute a challenging field of research. Sentence ordering is required for generation of coherent summaries. Theoretical analysis of the most common content ordering techniques have been provided but without any empirical study. Investigation of these techniques in more detail is an interesting direction in future work.

4. When looking at summaries generated by top DUC participating systems it becomes obvious that some sentence realization techniques have been applied, in particular sentence simplification. Making sentences more concise while preserving the important information can potentially improve precision and recall scores measured by ROUGE. Perhaps the next feature that should be added to the implemented framework, which can substantially improve its performance, is sentence simplification.

5. Automatic evaluation of summaries is an important but a non-trivial matter. The available methods such as ROUGE or Basic Elements are not very robust indicators of the quality of summaries. One of the recent TAC tasks directly aimed to develop automatic evaluation techniques. For example, it would be very useful to have a metric that assigns scores for content ordering or grammaticality of simplified sentences. Availability of such methods will greatly facilitate the development of the corresponding summarization components.

Bibliography

[1] C. Fellbaum, "WordNet: An electronic lexical database," *MIT Press*, vol. 12, 1998.

[2] T. Chklovski and P. Pantel, "Verbocean: Mining the web for fine-grained semantic verb relations," in *Proceedings of EMNLP*, vol. 4, no. Lin 1997, 2004, pp. 33–40.

[3] H. Dang and K. Owczarzak, "Overview of the TAC 2008 Update Summarization Task," in *Proceedings of Text Analysis Conference*, 2008, pp. 1–16.

[4] H. Dang, "TAC 2010 Guided Summarization Task Guidelines," 2010. [Online]. Available: http://www.nist.gov/tac/2010/Summarization/Guided-Summ.2010.guidelines.html

[5] C. Silla Jr and C. Kaestner, "An analysis of sentence boundary detection systems for English and Portuguese documents," in *Computational linguistics and intelligent text processing: 5th international conference, CICLing 2004, Seoul, Korea, February 15-21, 2004: proceedings*. Springer-Verlag New York Inc, 2004, p. 135.

[6] D. Palmer and M. Hearst, "Adaptive multilingual sentence boundary disambiguation," *Computational Linguistics*, vol. 23, no. 2, pp. 241–267, 1997.

[7] D. Gillick, B. Favre, D. Hakkani-Tur, B. Bohnet, Y. Liu, and S. Xie, "The ICSI/UTD Summarization System at TAC 2009," *dgillick.com*, 2009.

[8] T. Kiss and J. Strunk, "Unsupervised Multilingual Sentence Boundary Detection," *Computational Linguistics*, vol. 32, no. 4, pp. 485–525, Dec. 2006.

[9] L. Wenjie, W. Furu, L. Qin, and H. Yanxiang, "PNR 2 : Ranking Sentences with Positive and Negative Reinforcement for Query-Oriented Update Summarization," *Computational Linguistics*, no. August, pp. 489–496, 2008.

[10] P. Hu and D. Ji, "WHUSUM: Wuhan University at the Update Summarization Task of TAC 2009," *nist.gov*, 2009.

[11] T. Dunning, "Accurate methods for the statistics of surprise and coincidence," *Computational linguistics*, vol. 19, no. 1, p. 74, 1993.

[12] C. Manning and H. Schutze, *Foundations of statistical natural language processing*. MIT Press, 2000, vol. 78, no. 3.

[13] C. Lin and E. Hovy, "The automated acquisition of topic signatures for text summarization," in *Proceedings of COLING*, 2000, pp. 495–501.

[14] C. Tan, Y. Wang, and C. Lee, "The use of bigrams to enhance text categorization," *Information Processing & Management*, vol. 38, no. 4, pp. 529–546, 2002.

[15] A. Purandare and T. Pedersen, "Unsupervised word sense discrimination by clustering similar contexts," *University of Minnesota (August 2004)*, 2004.

[16] A. Newell, *The knowledge level*. Department of Computer Science, Carnegie-Mellon University, 1981.

[17] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[18] Z. Harris, "Distributional structure," *Word*, vol. 10, no. 23, pp. 146–162, 1954.

[19] J. Steinberger and K. Ježek, "Text summarization and singular value decomposition," *Advances in Information Systems*, pp. 245–254, 2005.

[20] J. Steinberger and M. Krišt'an, "Lsa-based multi-document summarization," in *Proceedings of 8th International Workshop on Systems and Control*, 2007, pp. 1–5.

[21] M. Sahlgren, "An introduction to random indexing," in *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005*. Citeseer, 2005, pp. 1–9.

[22] N. Chatterjee and S. Mohan, "Extraction-based single-document summarization using random indexing," in *19th IEEE International Conference on Tools with Artificial Intelligence, 2007. ICTAI 2007*, vol. 2, 2007.

[23] S. Ye, L. Qiu, T. Chua, and M. Kan, "NUS at DUC 2005: Understanding documents via concept links," *Proceedings of Document Understanding Conferences*, 2005.

[24] T. Copeck, A. Kennedy, M. Scaiano, D. Inkpen, and S. Szpakowicz, "Summarizing with Rogets and with FrameNet," in *The Second Text Analysis Conference*, 2009.

[25] K. Svore, L. Vanderwende, and C. Burges, "Enhancing single-document summarization by combining RankNet and third-party sources," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, no. June, 2007, pp. 448–457.

[26] V. Nastase, D. Milne, and K. Filippova, "Summarizing with Encyclopedic Knowledge," *h-its.org*, 2009.

[27] M. Wu and H. Kong, "Investigations on Event-Based Summarization," *Computational Linguistics*, no. July, pp. 37–42, 2006.

[28] E. Filatova and V. Hatzivassiloglou, "Event-based extractive summarization," in *Proceedings of ACL Workshop on Summarization*, vol. 111, 2004.

[29] M. Yoshioka and M. Haraguchi, "Multiple news articles summarization based on event reference," *Working Notes of the Fourth NTCIR Workshop Meeting,*, no. June, pp. 2–4, 2004.

[30] D. Marcu, *The theory and practice of discourse parsing and summarization*. MIT Press, 2000.

[31] C. Sporleder and A. Lascarides, "Exploiting linguistic cues to classify rhetorical relations," *Recent Advances in Natural Language Processing IV: Selected Papers from RANLP 2005*, vol. 157, 2007.

[32] K. McKeown, J. Klavans, V. Hatzivassiloglou, R. Barzilay, and E. Eskin, "Towards multidocument summarization by reformulation: Progress and prospects," in *The National Conference On Artificial Intelligence*, vol. pages. John Wiley Sons Ltd, 1999, pp. 453–460.

[33] J. Carbonell and J. Goldstein, "The use of MMR, diversity-based reranking for reordering documents and producing summaries," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, vol. pp. ACM, 1998, pp. 335–336.

[34] R. McDonald, "A study of global inference algorithms in multi-document summarization," *Advances in Information Retrieval*, pp. 557–564, 2007.

[35] A. Nenkova and L. Vanderwende, "The impact of frequency on summarization," *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005-101*, 2005.

[36] D. Radev, "Lexrank: graph-based centrality as salience in text summarization," *Journal of Artificial Intelligence Research*, vol. 22, 2004.

[37] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," in *Proceedings of EMNLP*. Barcelona: ACL, 2004, pp. 404–411.

[38] X. Zhu, A. Goldberg, J. Van Gael, and D. Andrzejewski, "Improving diversity in ranking using absorbing random walks," in *Proceedings of NAACL HLT*, 2007, pp. 97–104.

[39] H. Jing, "Summary generation through intelligent cutting and pasting of the input," 1998.

[40] C. Lin and E. Hovy, "Neats: A multidocument summarizer," in *Proceedings of the Document Understanding Workshop (DUC)*. Citeseer, 2001, pp. 6–9.

[41] R. Barzilay, N. Elhadad, and K. McKeown, "Inferring strategies for sentence ordering in multidocument news summarization," *Journal of Artificial Intelligence Research*, vol. 17, no. 1, pp. 35–55, 2002.

[42] M. Lapata, "Probabilistic text structuring: Experiments with sentence ordering," in *Proceedings of ACL*, vol. 3, 2003, pp. 545–552.

[43] J. Donghong and N. Yu, "Sentence Ordering based on Cluster Adjacency in Multi-Document Summarization," in *The Third International Joint Conference on Natural Language Processing*, 2008, pp. 745–750.

[44] J. Conroy, J. Schlesinger, and JG, "CLASSY query-based multi-document summarization," *Proceedings of the 2005*, 2005.

[45] L. Vanderwende, H. Suzuki, and C. Brockett, "Microsoft Research at DUC2006: Task-focused summarization with sentence simplification and lexical expansion," in *Proceedings of DUC*. Citeseer, 2006, pp. 70–77.

[46] D. Dunlavy, J. Conroy, J. Schlesinger, S. Goodman, M. Okurowski, D. OLeary, and H. van Halteren, "Performance of a three-stage system for multi-document summarization," *Federal Register*, vol. 1, no. H8, 2003.

[47] R. Barzilay and K. R. McKeown, "Sentence Fusion for Multidocument News Summarization," *Computational Linguistics*, vol. 31, no. 3, pp. 297–328, Sep. 2005.

[48] R. Passonneau, A. Nenkova, K. McKeown, and S. Sigelman, "Applying the pyramid method in DUC 2005," in *Proceedings of the Document Understanding Conference (DUC 05), Vancouver, BC, Canada*, 2005.

[49] C. Lin, "Rouge: A package for automatic evaluation of summaries," in *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, 2004, pp. 25–26.

[50] E. Hovy, C. Lin, L. Zhou, and J. Fukumoto, "Automated summarization evaluation with basic elements," in *Proceedings of the Fifth Conference on Language Resources and Evaluation (LREC)*. Citeseer, 2006, pp. 899–902.

[51] D. Lin, "A dependency-based method for evaluating broad-coverage parsers," *Natural Language Engineering*, vol. 4, no. 2, pp. 97–114, Jun. 1998.

[52] L. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1979.

[53] P. Bonacich, "Factoring and weighting approaches to status scores and clique identification," *Journal of Mathematical Sociology*, vol. 2, no. 1, pp. 113–120, 1972.

[54] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[55] J. Bollen, M. Rodriquez, and H. V. D. Sompel, "Journal status," *Scientometrics*, 2006.

[56] R. Navigli and M. Lapata, "An Experimental Study of Graph Connectivity for Unsupervised Word Sense Disambiguation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–14, 2009.

[57] J. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 604–632, 1999.

[58] B. Pittel, J. Spencer, and N. Wormald, "Sudden emergence of a giant k-core in a random graph," *Journal of Combinatorial Theory Series B*, vol. 67, pp. 111–151, 1996.

[59] J. Liu and S. Lee, "Study on feature select based on coalitional game," in *Neural Networks and Signal Processing*, 2008, pp. 445–450.

[60] N. Suri and Y. Narahari, "Determining the top-k nodes in social networks using the Shapley value," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, 2008, pp. 1509–1512.

[61] J. Conroy, J. Schlesinger, J. Goldstein, and DP, "Left-brain/right-brain multi-document summarization," *Proceedings of the*, 2004.