ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

Topic-based Amharic Text Summarization

By: Eyob Delele Yirdaw

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES OF THE ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

March 2011

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

Topic-based Amharic Text Summarization

By: Eyob Delele Yirdaw

ADVISOR:

Dr. Dejene Ejigu

APPROVED BY

EXAMINING BOARD:

1. Dr. Dejene Ejigu, Advisor _____

2. _____ _____

3. _____ _____

Acknowledgements

I would like to thank my advisor Dr. Dejene Ejigu for his unreserved efforts towards my work. He has been fully supportive from day one till the end. I greatly appreciate his comments which were enlightening and at times frustrating because I had to do a lot ☺

I am very grateful to Dr. Million Meshesha for his wholehearted support during the early days of title selection. Both he and my advisor were instrumental in giving the final shape to my thesis problem. I really appreciate it.

I am deeply indebted to Melese Tamiru for providing the code of his Amharic text summarizer. I am also grateful for the many interesting discussions we had together. Thanks so much for giving me what I needed on a silver plate. That is probably an understatement.

My sincerest gratitude goes to my family who were always there when I needed them. I am very fortunate to have had their full support throughout my work.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

CSF	Context-Sensitive Frequency-Based Score
CTM	Correlated Topic Model
CWS	Conditional Word Sentence
CWTS	Conditional Word Topic Sentence
DUC	Document Understanding Conference
EM	Expectation Maximization
HITS	Hyperlink-Induced Topic Search
ICT	Information and Communication Technology
IDF	Inverse Document Frequency
JS	Jensen-Shannon
JWS	Joint Word Sentence
JWTS	Joint Word Topic Sentence
KITS	Keywords in Topic Simplex
KIVSM	Keywords in Vector Space Model
KL	Kullback Leibler
LDA	Latent Dirichlet Allocation
Lp	Predictive Log-Likelihood
LSA	Latent Semantic Analysis
MMR	Maximum Marginal Relevance
NIST	National Institute of Standards and Technology
PAM	Pachinko Allocation
PHITS	Probabilistic HITS
PLSA	Probabilistic Latent Semantic Analysis
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
RST	Rhetorical Structure Theory
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TAC	Text Analysis Conference
TEM	Tempered Expectation Maximization
TF-IDF	Term Frequency – Inverse Document Frequency
VSM	Vector Space Model

Abstract

Automatic text summarization is important in today's information age where vast amount of information are produced for consumption. The case of Ethiopia is not an exception. The country has seen steady growth in digital content, ready for consumption by the mass. Compared to other international languages, text summarization works in Ethiopia's local languages in general and the Amharic language in particular, can be said to be in their early stages of development. In this regard, more work should be carried out to meet present and future needs of the availability of high quality information that needs to be extracted from large collections of data in a timely manner.

This thesis investigates the problem of building a concept-based single-document Amharic text summarization system. Because local languages like Amharic lack extensive linguistic resources, we propose to use statistical approaches called topic modeling to create our text summarizer. The proposed algorithms are language and domain independent and hence can also be used for other local languages. More specifically, we propose to use the topic modeling approach of probabilistic latent semantic analysis (PLSA).

We show that a principled use of the term by concept matrix that results from a PLSA model can help produce summaries that capture the main topics of a document. We propose six algorithms to help explore the use of the term by concept matrix. All of the algorithms have two common steps. In the first step, keywords of the document are selected using the term by concept matrix. In the second step, sentences that best contain the keywords are selected for inclusion in the summary. To take advantage of the kind of texts we experiment with (news articles) the algorithms always select the first sentence of the document for inclusion in the summary.

We evaluated the proposed algorithms for precision/recall for summaries of 20%, 25% and 30% extraction rates. The best results achieved are as follows: 0.45511 at 20%, 0.48499 at 25% and 0.52012 at 30%. We also compared our systems with previous summarization methods that have been developed for other languages based on topic modeling approaches using our Amharic data set. Our results show that the proposed algorithms perform better at all extraction rates.

Keywords: Amharic Text Summarization, Keyword Approach, Probabilistic Latent Semantic Analysis.

Chapter 1. Introduction

1.1. Background

People in various areas have become increasingly exposed to vast amount of information. Their work usually involves filtering the main substance out of large amount of potentially relevant information. However, given the reality of time constraint to process the available information and make the best out of it, it is very wise to think of computer assisted automatic text summarization as a solution.

In Ethiopia, currently, there is huge amount of digital data being available in Amharic. Many organizations and individual users are in need of efficient Amharic text processing tools like retrieval, classification and summarization systems. Access to automatically produced quality summaries helps in increasing organizational/personal productivity by supporting timely information access.

The beginnings of automatic text summarization dates back to 1958 in Luhn's work (Luhn 1958). Currently, two types of summarization techniques are employed: abstractive and extractive (Jones 2007). Abstractive summarization tries to convey summarized information by taking the ideas from the most important sections of the document and paraphrase it, possibly including new words that did not appear in the original document. However, text summarization in this manner involves natural language understanding and generation systems and these are yet developing fields. Extractive text summarization systems try to identify those sentences that are considered most important to the user and present it as they are. It is this technique that has found the most attention in summarization research than the abstractive one. This work focuses on extractive summarization techniques.

Summarization can also be classified into single-document and multi-document summarization. In single-document summarization, we generate summary of only one document. In the case of multi-document summarization, summary is generated for a collection of documents. These

documents can, for instance, be the output from a search engine. In this thesis, we have experimented with single-document summarization.

Text summarization techniques range from shallow statistical analysis to deep linguistic techniques. Common ones include simple statistical methods such as the ones based on term frequency, graph-based approaches and more recently machine learning approaches such as support vector machine (SVM) and latent semantic analysis (LSA) (Jones 2007).

Various research works in English document summarization have shown that systems that use some form of conceptual analysis achieve higher scores in general when compared to systems that do not (Jones 2007). A number of works exist that apply concept-based approaches to summarization of English documents. Steinberger et al. (Steinberger et al. 2007) make use of anaphoric resolution and LSA to build their text summarization system. They compare their work with the Document Understanding Conference[1] (DUC 2002) evaluation results and achieve the second best score. Harabagiu and Lacatusu (Harabagiu and Lacatusu 2002) participated in the DUC 2002 single and multi-document summarization tasks by using a number of language specific features such as named-entity recognizer, phrasal parser and entity co-reference. They were among the top two teams in the overall evaluation scores of the summarization tasks. Hennig (Hennig 2009) applies probabilistic latent semantic analysis (PLSA) to query focused multi-document summarization. Their system outperforms the best systems of DUC 2006.

A number of works have been done on the summarization task of Amharic documents. However, all but one of these works (see for example Kamil 2004; Abraham 2007; Teferi 2005) are not concept-based. Kamil (Kamil 2004) and Abraham (Abraham 2007) employ surface level statistical features to summarize Amharic text. Teferi (Teferi 2005) uses Naïve Bayes machine learning approach based on four features (presence of title words, location of sentences in the document, presence of cue words and presence of thematic or highly frequent words). Melese (Melese 2009) has used an LSA approach to address this gap to single-document summarization. However, LSA has a drawback, namely its unsatisfactory statistical foundations (Hofmann 1999b). The development of a high performance summarizer is quite a challenging task.

[1] www-nlpir.nist.gov/projects/duc

Effective concept-based solutions need to be applied to provide satisfactory solutions to this problem.

The main aim of this thesis work is to build upon previous Amharic text summarization works. We do this by investigating the application of language independent statistical methods called topic modeling for concept-based single-document Amharic text summarization. The topic modeling approaches we investigate are LSA, PLSA and latent Dirichlet allocation (LDA).

LSA is an automatic algebraic-statistical technique that can be used to extract and represent the contextual usage of words' meanings in textual documents (Steinberger 2007). The method first constructs a term by sentence matrix. Then, singular value decomposition (SVD) is applied to the matrix. This results in the derivation of the latent semantic structure of the document represented by the matrix. This is equivalent to breaking the original document into a number of linearly independent base vectors that express the main topics of the document. SVD enables terms and sentences to be clustered on a semantic basis rather than on the basis of words only.

As mentioned previously LSA has a number of drawbacks mainly due to its unsatisfactory statistical foundations. LSA assumes that words and documents form a joint Gaussian model. However, the observed distribution is Poisson (Roelleke 2003). Probabilistic latent semantic analysis (PLSA), which is the statistical version of LSA, is based on a multinomial model. PLSA and is reported to give better results than standard LSA (Hofmann 1999b; Bhandari et al. 2008). LDA is also a probabilistic topic modeling approach like PLSA. It further improves upon PLSA and has shown performance gains over PLSA in information retrieval experiments (Blei et al. 2003).

1.2. Objectives

The general and specific objectives of the research are described below.

1.2.1. General Objective

The general objective of the research is to investigate the application of concept-based approaches to Amharic document summarization and propose an improved system.

1.2.2. Specific Objectives

With the aim of achieving the general objective of the study, the following specific objectives are identified:

- Conduct literature review on the various ways of using concept-based approaches to document summarization and in particular the topic modeling approaches of LSA, PLSA, and LDA.
- Propose concept-based summarization algorithms that are language and domain independent.
- Develop a prototype Amharic text summarization system based on the proposed algorithms.
- Evaluate the performance of the developed summarization system.

1.3. Methodology

The solution strategy to be adopted for the research is described as follows.

1.3.1. Literature Review

Extensive literature review will be conducted on concept-based approaches that could be used for Amharic document summarization. Particular focus will be given to techniques of topic modeling and summarization works that use these statistical methods. Existing Amharic document summarization systems will also be reviewed thoroughly.

1.3.2. Data Collection

Data necessary for the purpose of experimentation will be acquired after assessing the different options that are available. Some of the options include online news paper articles, news sites and media organizations such as news agencies. After the data is collected, it will be annotated and given to three independent human evaluators. The human evaluators will generate model summaries against which the system will be evaluated.

1.3.3. Design and Implementation

After literature review has been carried out and based on the identified research problem, we try to design the summarization algorithms. The Java programming language with the NetBeans[2] environment will be used to help code the summarizer. We also carry out parallel implementations of our algorithms with Matlab to validate our implementations of the algorithms with Java. Matlab is preferred since it can compactly represent calculations that take quite a number of lines in other programming languages.

1.3.4. Experimental Evaluation

In order to test the performance of the developed system, it will be compared with manual summaries generated by the human. The summary generated by the prototype system will be evaluated using precision, recall and F-measure. We also plan to implement text summarization algorithms from the literature and compare their performance with the algorithms that we propose.

1.4. Expected Contributions

We believe that the outcome of the research is crucial in many respects

- It enhances existing approaches that do not follow conceptual analysis to Amharic text summarization.

[2] http://www.netbeans.org

- It contributes towards the realization of robust Amharic summarization systems which should ultimately benefit from the principles of conceptual analysis.
- Various organizations/individuals benefit from the resulting research work which can be considered as a step towards producing high-performance Amharic text summarization system.

1.5. Thesis Outline

This thesis is organized in five chapters. Chapter 2 presents a detailed discussion of the literature concerning text summarization. The chapter discusses basic notions of text summarization, how automatically generated summaries are evaluated, discusses three topic modeling techniques (LSA, PLSA and LDA), reviews approaches to text summarization systems and finally states the research problem of the thesis. Chapter 3 presents the proposed system for single-document Amharic text summarization. In Chapter 4, we discuss the design of the experiments carried out and also present the results along with discussions of the outcome. Finally, Chapter 5 gives conclusion to the work carried out and tries to suggest future directions for further research.

Chapter 2. Literature Review

This chapter discusses state-of-the-art in the field of automatic text summarization. Basic notions of text summarization such as summary types and stages of automatic text summarization are discussed in section 2.1. All automatic systems need some form of evaluation. In light of this, section 2.2 discusses two of the widely used objective evaluation measures for text summarization systems, co-selection and content-based summary evaluations. Section 2.3 focuses on the three state-of-the-art dimensionality reduction methods, LSA, PLSA and LDA. Although these methods use pure lexical methods for topic identification, they have very competitive performance when compared to other summarization systems that employ elaborate linguistic knowledge/features. They are particularly attractive because they are language and domain independent. Section 2.4 discusses various approaches to text summarization. More focus is given to approaches that use topic modeling (latent variable models) as they are the focus of our work. Finally, section 2.5 describes the gap identified and the research problem of the thesis.

2.1. Basic Notions of Text Summarization

The first automatic summarization system was invented by (Luhn 1958). It was in the hope that automatic summaries that are generated without requiring the expensive intellectual efforts of human beings would greatly help in quickly satisfying the information needs of a user. Mani (Mani 2001) defines summarization as *taking an information source, extracting content from it, and presenting the most important content to the user in a condensed form and in a manner sensitive to the user's or application's needs*. Since Luhn's pioneering work, the field of automatic text summarization has grown to a much richer field, with different system approaches and summarization tasks. This section provides introductory material on text summarization.

2.1.1. Types of Summarization

Automatic text summarization can be categorized into a number of classes such as abstractive and extractive, generic and query-based or single document, multi-document and update summarization. Each of these classes can be combined to define a summarization task.

Abstractive and Extractive

When humans are asked to create a summary for a given document, they typically form abstractive summaries. An abstractive summarization process is summarization where the summarizer (human or system) extracts the most important points of a document and presents the information by applying the necessary modifications to the original sentences of the document to produce a coherent and readable text (Radev et al. 2002). Abstractive summarization is typically characterized by using a number of new words not present in the original document. In contrast, extractive summarization is characterized by the selection of whole sentences from the document to be summarized such that they would contain the main points of the document (Radev et al. 2002).

Generic and Query-based

Generic text summarization creates summaries that provide an overall sense of the document's contents (Gong and Liu 2001). This amounts to selecting contents that correspond to the main topics of the document. Creating a generic text summarization system can be regarded as extracting information from the document that the author considers the most important (Bosma 2008). Query-based summarization focuses on a user of information than the author of the document in that it generates summary based on the user's need. The user's information need is basically provided in the form of query. Query-based summarization has become especially important as the amount of information online is growing exponentially (Carbonell and Goldstein 1998).

Single Document, Multi-document, and Update Summarization

The sources of documents can be a factor to decide which of the three types of summarization is performed. If a user is interested in only the contents of a single document, perhaps consisting of

a number of pages, we have single document summarization. However, a user may not be able to get his information needs from a single document. If a user wants comprehensive information on a certain topic, it is quite likely that a single document would not provide all the required information. In such a case multiple documents selected by the user would be given to a multi-document summarization system. In case the user wants to update the information that s/he has, say at a later date, with a new set of documents, then this task should be performed by an update summarization system. The update summarization system is assumed to have access to the previous set of documents summarized for the user. Update summarization task has been introduced in year 2008 of the Text Analysis Conference (TAC) organized by NIST. Multi-document summarization (update summarization) is more complex than single-document summarization (Jones 2007). Many issues that do not appear in single-document summarization become crucial in multi-document summarization, some of the challenges being avoiding redundancy and ordering of sentences in the final summary.

2.1.2. Stages of Automatic Text Summarization

Researchers in automatic text summarization have identified three stages: topic identification, interpretation and summary generation (Hovy and Lin 1999; Hovy 2005). Each of the three stages is performed sequentially in the given order above to create a summary.

Topic identification involves the identification of central or important topics from the document to be summarized. This stage should decide the units that represent the topics of the document. This could be words, phrases, sentences or larger units such as paragraphs. After identification of these units, the next step is scoring the units based on their importance. Finally, the top scoring units are selected to represent the important topics of the document.

The interpretation stage involves the analysis of the topics to understand how they are related, such as interpreting some concepts as encompassing some others (Hovy and Lin 1999). This stage is necessary because listing out text units that resulted from the topic identification stage will not form a good summary. An abstractive text summarization system should perform this step as opposed to an extractive system which does not include this step. By definition, an

extractive summarizer does not modify the original sentences of the document in ways that are required at this stage. This stage necessitates important steps such as concept fusion and evaluation.

The generation phase accepts the output of the interpretation stage. Output from the interpretation stage is generally made from bits and pieces of information that have been discovered at that stage and hence need further processing before being presented to humans. The generation phase is responsible for combining this information to create a coherent and readable output.

2.2. Summary Evaluations

Different approaches exist to evaluate the performance of automatic text summarization systems. We follow the classification scheme of (Radev et al. 2003; Steinberger 2007). The authors classify the evaluation approaches into the broad categories of intrinsic and extrinsic (task-based). The intrinsic evaluation involves assessing the quality of a summary by comparing it to an ideal summary (summary produced by humans). The extrinsic evaluation involves assessing the quality of a summary by applying it to a certain task such as document categorization, question answering or information retrieval.

The intrinsic evaluation of text summaries can be of two types: text quality evaluation and content evaluation. Text quality evaluation involves assessing the system summary for various linguistic features such as grammatical correctness, non-redundancy, referential clarity, and structure and coherence. The most widely used approach for summary evaluation is the content evaluation approach. There are two types of content evaluation approaches: co-selection and content-based. Co-selection evaluation methods are suitable for summaries produced by sentence selection from the document to be summarized and where the ideal summaries are made from sentence extracts from the original document. These include precision, recall and F-measure. Content-based measures are suitable for comparing system summaries with human abstracts that may contain new words that are not found in the original document. However, they can still be used to evaluate summaries where the ideal summaries are extract-based. Content-based

measures include cosine similarity and ROUGE. Co-selection and content-based evaluations are discussed in some detail in the following subsections as they are the most relevant ones to our work.

2.2.1. Co-selection Summary Evaluations

Co-selection evaluation measures are borrowed from information retrieval. They are defined as follows (equations 2.1, 2.2 and 2.3) (Nenkova et al. 2006; Steinberger 2007).

Recall (R) gives the fraction of sentences that the system has chosen from the totality of sentences found in the ideal summary.

$$R = \frac{|system\ and\ human\ choice\ overlap|}{|sentences\ chosen\ by\ human|} \tag{2.1}$$

Precision (P) measures the fraction of system summaries that are correctly chosen.

$$P = \frac{|system\ and\ human\ choice\ overlap|}{|sentences\ chosen\ by\ system|} \tag{2.2}$$

F-score (F) is the harmonic mean of recall and precision.

$$F = \frac{2 * P * R}{P + R} \tag{2.3}$$

The co-selection methods discussed here are perhaps the simplest of all evaluation measures. This simplicity comes at a price though. The main problem is the difficulty in accounting for variations in what humans consider ideal summary sentences. Human summarizers do not often agree on what constitutes an ideal summary (Donaway et al. 2000). Thus, a summarization system may select good sentences but get low scores based on these evaluation measures. This can be mitigated by having more than one human summarizer generate an ideal summary and using the resulting ideal summaries in a number of different ways such as majority voting,

intersection or average of results of evaluation measures obtained from each of the ideal summaries. Numerous content-based measures, such as the cosine similarity and ROUGE, have been proposed to improve upon the drawback of co-selection measures.

2.2.2. Content-based Summary Evaluations

Content-based measures enable to evaluate summaries that may not contain exact sentences as the ideal summaries. Exactness would be overly restrictive because two sentences might be written differently but contain more or less similar information.

Cosine Similarity

The cosine similarity uses the vector space model to represent both the system and ideal summaries (see next section for discussion of the vector space model). The axes of the vector space are words contained in either summary. Hence, a summary can be represented as a point in this vector space. To compare the similarity of two summaries, one measures the cosine of the angle between the vectors that represent the summaries. If we represent the system summary by S and the ideal summary by I, the cosine similarity between the two is given by

$$\cos(S, I) = \frac{\sum_k S_k * I_k}{\sqrt{\sum_k (S_k)^2} * \sqrt{\sum_k (I_k)^2}} \quad (2.4)$$

ROUGE

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It uses n-gram co-occurrence statistics for summary evaluations and has shown to correlate well with human evaluations (Lin 2004). It has now become the most widely used evaluation measure. There are a number of measures that are defined by the ROUGE package such as ROUGE-n, ROUGE-L and ROUGE-SU4.

ROUGE-n, where n is a positive integer, is given by equation 2.5 below

$$\frac{\sum_{S \in \{Reference\ Summaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{Reference\ Summaries\}} \sum_{gram_n \in S} Count(gram_n)} \quad (2.5)$$

Here, n stands for the length of the n-gram used and $Count(gram_n)$ stands for the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries.

2.3. Dimensionality Reduction

This section discusses the state-of-the-art statistical techniques that are used for dimensionality reduction. These statistical techniques and their extensions have become very popular in a number of areas like information retrieval, sentiment analysis, collaborative filtering, text categorization, and text summarization. Due to time limitations, our treatment will be restricted to discussions of latent semantic analysis (LSA), probabilistic latent semantic analysis (PLSA) and latent Dirichlet allocation (LDA). There are many more new models that improve upon these such as the correlated topic model (CTM) (Blei and Lafferty 2005) and pachinko allocation (PAM) (Li and McCallum 2006). Except LSA, all statistical techniques mentioned here are called probabilistic topic models.

2.3.1. Vector Space Model

We will start with a discussion of the vector space model (VSM) (Salton et al. 1975) which is the starting point for LSA.

VSM was first introduced for use in information retrieval. It is a means of indexing documents based on term frequencies. It starts with identification of unique indexing terms. These terms are usually filtered with a list of stop-words (words that occur in many documents and hence are statistically unhelpful to estimate importance of a document). Each unique term serves as a single dimension, resulting in an m-dimensional space, where m is the number of unique terms. A document can then be represented by means of co-ordinates of a point in this space. The co-ordinates for a document are actually the frequencies of the words (terms) that it contains. This space can also be represented by an $m \times n$ matrix where n refers to the number of documents. Once we have this representation, the next step is term weighting. Term weighting is necessary

to indicate the importance of a term in representing a document. A widely used weighting scheme is the *tf-idf* (term frequency, inverse document frequency). This gives us a representation of each document by a vector of terms where each term is associated with a weight w_{ik} determined as given in equation 2.6 below

$$w_{ik} = tf_{ik} \times \log\left(\frac{N}{n_k}\right), \tag{2.6}$$

where tf_{ik} is the number of occurrences of term T_k in document D_i, N is the number of documents in a given collection, and n_k represents the number of documents containing term T_k. Because terms that appear in many documents do not usually contribute in discriminating against documents, the *idf* score is introduced. The *idf* score gives less weight to terms that appear in many documents. Similarity of a query to a document or similarity of documents with one another is calculated by the use of cosine similarity (see equation 2.4). Note that because VSM and the other latent variable models were introduced for information retrieval, we often talk of a term by document matrix. For text summarization, this usually becomes a term by sentence matrix.

2.3.2. Latent Semantic Analysis

Although VSM has been successfully used in modern Internet search engines, it has drawbacks. These problems have been investigated at length by the inventors of LSA (Deerwester et al. 1990). Its problems come as the result of the high dimensionality associated with the model. A VSM with m indexing terms is m-dimensional. This leads to reduced performance of an information retrieval system based on VSM. The reduced performance comes as a result of the phenomena of synonymy and polysemy in natural languages. Synonymy refers to two words having the same/similar meaning while polysemy refers to a word that could be used in different contexts with different meanings. The problem with VSM results from the fact that it indexes documents by words that are found only in the document. A simple example can illustrate why indexing in this way can lead to reduced performance. Let us say we have four documents *A*, *B*, *C* and *D* and a user query *Q*. Document *A* is about information retrieval and contains the word *retrieve*. Document *B* is about information retrieval and does not contain the word *retrieve* but

contains the word *access*. Document *C* is also about information retrieval and contains both of the words *retrieve* and *access*. Document *D* is not about information retrieval but about retrieving lost documents from formatted hard disks and contains the word *retrieve*. *Q* is a user query represented by *information retrieval*. When the user searches with his query, *Q*, only documents *A*, *C*, and *D* are returned because they contain the matching word *retrieval*. The problem of polysemy has resulted in document *D* being returned, while the problem of synonymy has prevented document *B* from being returned. This shows that document *B* should have been indexed not only by the word it contains, *access*, but also by the word *retrieve*, which has similar meaning with the former word in this context. Document *D* should have been indexed by additional terms that give it a representation far away (e.g. measured by cosine similarity) from the rest of relevant documents based on the terms it contains. This gives us a representation of documents by their meaning, concept or topic. Such a space can thus be called a semantic space.

The authors of LSA proposed an approach to automatically index documents in a semantic space. The method they use is a statistical method. It makes use of only co-occurrence counts (co-occurrence refers to a word occurring in a document with another word) obtained from the VSM model. We can use the above example to illustrate the basic method. Documents *A*, *B* and *C* are all related and talk about the same subject (topic or concept), identified by the presence of words *access* and *retrieval*. Because both *access* and *retrieval* occur in document *C* which is about information retrieval, we add the words *access* and *retrieval* to all documents that talk about information retrieval that do not contain the words; in our case documents *A* and *B*. Thus querying with *Q* should now return document *B*.

LSA achieves such form of semantic indexing by the use of singular value decomposition (SVD), a technique from linear algebra. The authors point out that SVD is not limited to discovering simple pair-wise correlations, as illustrated by the above simple example. It is able to discover implicit higher-order structures in the association of terms with documents. The method is termed *latent* because these correlations are hidden from the original VSM representation model. What SVD does is find a subspace (called latent semantic space or simply latent space) in the original VSM space, where the subspace now has less number of dimensions but captures

most of the variance of the original data (Blei et al. 2003). This can be achieved by following these iterative steps. Start with an axis from the original VSM space and rotate it so that it is aligned to portion of the data where the variance is greatest. Next, find a new axis that is perpendicular to the first axis and orient it in a direction where it aligns with the greatest data variance available. The second step is repeated until we come up with a set of axis that can account for a majority of the variance of the original data. We end up with only a few dimensions, say 20% of the original, for indexing. The remaining dimensions of the new subspace are discarded because they tend to model noise in the data or randomness rather than a pattern. The remaining dimensions now represent the previously hidden major statistical patterns in the original data. In the words of (Deerwester et al. 1990), *roughly speaking, these factors may be thought of as artificial concepts; they represent extracted common meaning components of many different words and documents.*

We now describe the technical details of SVD. SVD starts with an $m \times n$ matrix A. A represents the data from a VSM space. m represents the number of unique terms and n the number of documents. Thus, a column in this matrix represents the co-ordinates of a document in a VSM space where the axes represent terms of the document. Alternately, if we create a VSM space where the axes represent the documents, then the rows of A represent coordinates of a term in this space. SVD decomposes matrix A it into three matrices U, V, and Σ

$$A = U \Sigma V^{T} \tag{2.7}$$

U is an $m \times c$ matrix, Σ a $c \times c$, and V an $n \times c$ matrix. c represents the number of artificial concepts, where $c = n$.The columns of U are called left singular vectors while the columns of V right singular vectors. Σ is a diagonal matrix and its values are called singular values. If the singular values of Σ are ordered in decreasing order, matrix A will have a unique set of singular values (though the matrices U and V are not). Both of the singular vectors form orthonormal vectors (vectors which are of unit length and perpendicular to each other). These orthonormal vectors are actually the axes of the new subspace. Let us see how we transform the data matrix A using these vectors.

If we start with a VSM where the axes are terms, we use the transformation of the matrix A via the left singular vectors to arrive at a *concept by document* representation. To get the required transformation expression, we simply multiply each side of equation 2.7 by U^T and get $U^T A = \Sigma V^T$. The rows of U^T act upon the columns of A to come up with new coordinates for the original documents of A. The new coordinates of the documents are now represented by columns of the matrix ΣV^T. The singular values in Σ represent the importance of the *concepts* that correspond to the rows of V^T as determined from the original data. Importance, as discussed above, is measured by the greatest variance (signal) in the data. Thus, V^T represents the un-scaled *concept by document* matrix. The scaling factors are the singular values represented by Σ. A similar line of argument leads us to conclude that U represents the un-scaled *term by concept* matrix. As a final step, concepts that have corresponding low singular values must be discarded. This can be achieved by setting the singular values that correspond to these concepts to zero. Hence, we now have a reduced representation of A, A' given by equation 2.8 below

$$A'_{m\times n} = U'_{m\times c'} \quad \Sigma'_{c'\times c'} \quad V'^{T}_{c'\times n} \tag{2.8}$$

c' in equation 2.8 is the number of significant (dominant or useful) concepts that have been retained so as to preserve the essential statistical features for various tasks like summarization, retrieval, and classification. While the original matrix A is sparse, i.e. has many zero entries, the new matrix A' is not. The zero entries of matrix A correspond to the absence of terms which should have been present in the original matrix for indexing. In the information retrieval example discussed above, this corresponds to documents *A* and *B* which should have included both of the terms *access* and *retrieval* to enable accurate document retrieval.

The discussions so far have focused on finding reduced dimensional representations of documents that are found in the original document collection. But we need to be able to represent new documents or queries in the latent space. The reason is that queries can then be compared to the existing documents in a reliable manner. The procedure that we follow to find representations for queries in the latent space is called "folding-in" or "projection". If we have a query Q with the following term counts, $Q = [t_1\ t_2\ \dots\ t_m]^T$, then the reduced dimension representation for the query becomes $q = Q^T U'$ (Deerwester et al. 1990). The folding-in process simply starts with the

VSM representation of the query $Q = [t_1\ t_2\ \ldots\ t_m]^T$ and projects it using orthogonal projection onto the subspace representing the latent semantic space. This amounts to finding the coordinates of q, along each axis of the subspace that represents the artificial concepts, by computing the magnitude of the vector represented by Q along each of the directions of the artificial concepts.

2.3.3. Probabilistic Latent Semantic Analysis

PLSA and LSA are similar in that they perform dimensionality reduction. Their difference lies in the manner in which they achieve the dimensionality reduction. LSA relies on SVD. SVD discovers a subspace of a matrix representing a VSM model by minimizing the sum of squared distances between the data points and a given set of orthonormal vectors. This is equivalent to making the assumption that words and documents form a joint Gaussian distribution (Hofmann 1999b). This Gaussian distribution assumption is the drawback of LSA. This assumption may not be problematic for continuous data but for discrete data such as text documents, the assumption is not expected to hold and may result in degraded system performance as demonstrated by Hofmann (Hofmann and Puzicha 1998; Hofmann 1999a; Hofmann 1999b; Hofmann 2001).

Hofmann introduces what he calls the aspect model (PLSA), which is as a generative statistical model for discrete data. The generative process can be described as follows:

- Select a document d, with probability $p(d)$,
- Pick a latent variable z, with probability $p(z)$,
- Generate a word w with probability $p(w|z)$.

This can be translated into a joint probability model

$$p(d,w) = p(d)p(w|d) \tag{2.9}$$

$$p(w|d) = \sum_z p(w|z)p(z|d) \tag{2.10}$$

$p(w|d)$ becomes a representation of a document. It is a probability distribution which is used to generate words of document d. z represents one of the latent variables (concepts/topics) of the model. This means that $p(w|z)$ gives us the probability of generating a word w given topic z. A given document usually talks about a number of topics. That is, each word of a document can be about a different topic. Some topics can be dominant in a document while others might not be. These proportions are represented by the multinomial distribution $p(z|d)$, and called mixing proportions for the multinomial distributions $p(w|z)$. Thus, $p(w|d)$ is composed of a mixture of probability distributions $p(w|z)$ with mixing weights $p(z|d)$ and hence is a mixture model.

Hofmann (Hofmann 1999b) gives a geometrical interpretation of the PLSA model using equation 2.10. Let us first define the *word simplex*[3] (Blei et al. 2003) as an $m-1$ dimensional space where all possible mutltinomial distributions of words, $p(w)$, are represented. The vertices of the word simplex correspond to a single word where $p(w)$ for that word is one while $p(w)$ for all other words is zero. Any point inside the word simplex has non-zero probability distribution for all of the words. The K conditional distributions of words given a topic, $p(w|z)$ can be represented as points in the word simplex, where K is the number of topics of the PLSA model. Via its convex hull, this set of K points defines a $K-1$ dimensional sub-simplex called the topic simplex (Blei et al. 2003).

Equation 2.10 shows us that the empirical distributions $p(w|d)$ for each document are approximated by a convex combination[4] of vertices of the topic simplex. The mixing proportions for a document $p(z|d)$ would then correspond exactly to the coordinates of the document in the topic simplex. This can be represented geometrically by figure 2.1 below

[3] An object is convex (convex set) if for every pair of points within the object, every point on the straight line segment that joins them is also within the object. The convex hull or convex envelope for a set of points in a vector space is the minimal convex set containing the set of points. An n-simplex is an n-dimensional polytope (a geometric object with flat sides) with $n+1$ vertices, of which the simplex is the convex hull.

[4] A convex combination is a linear combination of points where all coefficients are non-negative and sum up to 1. All possible convex combinations will be within the convex hull of the given points.

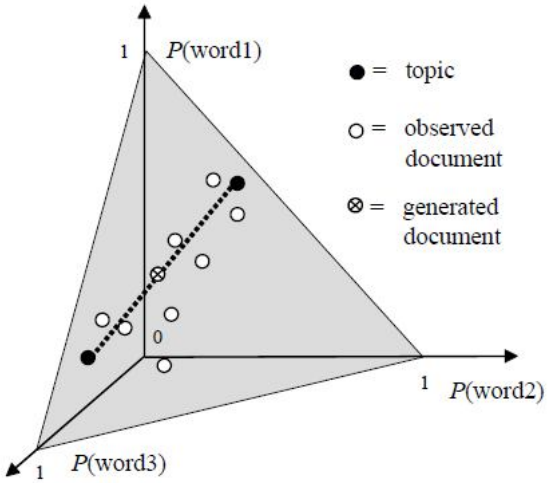

Figure 2.1: A geometric interpretation of the PLSA model. The black dot corresponds to the distribution $p(w|z)$ which characterizes a topic of the document collection. In this model, we have eight documents, three words and two topics. The word simplex is the triangle, while the dotted straight line segment which has the two black dots at its ends is the topic simplex. All objects (the words, topics, observed and generated documents) are located in the word simplex (the words located at the vertices of the word simplex). Figure is taken from (Steyvers and Griffiths 2006).

The generative model of PLSA can be represented by the following graphical model (figure 2.2):

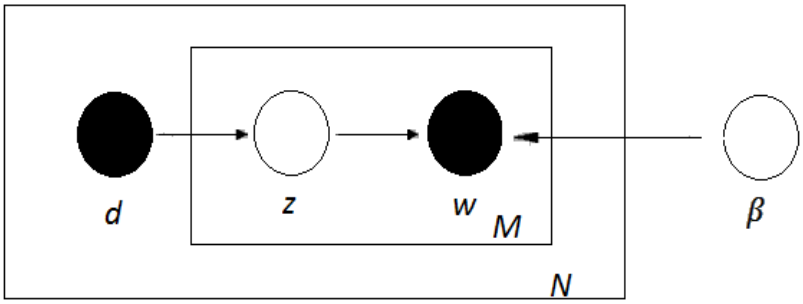

Figure 2.2: Graphical model of PLSA.

M is the number of words in a document and N is the number of documents in our collection. The variables d and w are represented by shaded circles because they are observable variables. β is parameter of the distribution $p(w|z)$. We have to solve a representation of this graphical

model for the parameters of our documents which are $p(w|z)$ and $p(z|d)$. This is achieved by the maximization of the log-likelihood of our observed data. A likelihood function is obtained from a probability distribution function (*pdf*) by treating the parameters of the *pdf* as the variables and the random variables of the *pdf* as constants. The log-likelihood is simply the logarithm of the likelihood function. It is suitable for the maximization process. By assuming that each observation pair (d, w) is generated independently of on another (the bag-of-words assumption), the likelihood function can simply be obtained by multiplication of $p(d, w)$ for all the observed pairs (d, w). It is given by

$$\mathcal{L} = \sum_{d \epsilon D} \sum_{w \epsilon W} n(d, w) \; log \; p(d, w) \tag{2.11}$$

where $n(d, w)$ denotes entry of the term by document matrix for a document d and a word w. For the maximization process, we substitute equations 2.9 and 2.10 in equation 2.11 for $p(d, w)$. Setting the partial derivatives of the log-likelihood with respect to the parameters to zero and solving the resulting equation will not yield a closed form solution. Instead one has to follow an iterative procedure for solving models with latent (unobservable) variables. This iterative procedure is called Expectation Maximization (EM). The procedure starts with random values of the parameters of the model, $p(w|z)$ and $p(z|d)$, and tries to update these values iteratively to find the values that maximize the function $\mathcal{L}$. The process involves two steps: expectation step (E-step) and Maximization step (M-step). In the E-step we calculate posterior probabilities of the latent variables in the model given each pair (d, w). In the M-step, the parameters of the system are updated for given posterior probabilities computed in the previous E-step. Alternating these two steps results in a convergent procedure that approaches a local maximum of the log-likelihood. For the E-step we use the equation

$$p(z|d, w) = \frac{p(w|z)p(z|d)}{\sum_{z'} p(w|z')p(z'|d)} \tag{2.12}$$

Following the M-step results in the re-estimation equations

$$p(w|z) = \frac{\sum_d n(d,w)p(z|d,w)}{\sum_{w'}\sum_d n(d,w')p(z|d,w')} \tag{2.13}$$

$$p(z|d) = \frac{\sum_w n(d,w)p(z|d,w)}{n(d)} \tag{2.14}$$

where $n(d) = \sum_w n(d,w)$ refers to the document length, i.e., the number of words in a document. The estimation of the parameter $p(d)$ is straightforward and does not need the EM algorithm. It can be estimated separately from the other parameters. It is given by

$$p(d) = \frac{n(d)}{N} \tag{2.15}$$

where $N = \sum_{d'} n(d')$ is the total number of words in the document collection.

The graphical model of figure 2.2 is asymmetric in w and d. A symmetric graphical model in w and d results in the parameters $p(z)$, $p(w|z)$, $p(d|z)$ (Hofmann 1999b). Both the asymmetric and symmetric graphical models are equivalent.

Hofmann observed two problems with his PLSA model (Hofmann and Puzicha 1998). First, the EM algorithm guarantees only a local maximum, whereas in reality there could be many local maxima. Second, the trained model may not generalize well to unseen data, such as new documents. He tried to solve these two problems by introducing tempered EM (TEM) algorithm. Without going into the details of the derivation, TEM introduces a modification to the E-step via a tempering parameter β as

$$p(z|d,w) = \frac{[p(w|z)p(z|d)]^\beta}{\sum_{z'}[p(w|z')p(z'|d)]^\beta} \tag{2.16}$$

When $\beta = 1$ we get the standard E-step. Tempering reduces the sensitivity of EM to local maxima and enables a better predictive model for unseen data.

Folding-in in a PLSA model for a query document q represented in the word simplex proceeds by computing the mixing proportions, $p(z|q)$, by EM iterations, where the factors $p(w|z)$ are fixed such that only the mixing proportions $p(z|q)$ are adapted in each M-step. Hofmann (Hofmann 1999b) shows that folding a query in this manner represents projecting the query into the topic simplex by minimization of the Kullback Leibler (KL) divergence[5] between the query and points of the topic simplex.

In summary, what PLSA does is, instead of fixing the probability distribution of the words and documents before hand, as is the case of LSA, we can start with an unknown distribution of words/documents with respect to a set of unobservable variables, called latent variables. These latent variables serve as representation of *concepts* in our model. This distribution will then be used to derive an expression for the likelihood function of the observed data. Finally, the parameters of this model are estimated by following the procedures of Expectation Maximization (EM).

2.3.4. Latent Dirichlet Allocation

The total number of parameters of a PLSA model is $km + kn$ for an $m \times n$ term by document matrix and k number of topics. Thus, we have a linear growth of the parameters of the model with the number of documents (training set). This leads to overfitting of the parameters of PLSA to the training data and lessens the predictive performance of the model to new documents. This problem persists even if tempering is used (Blei et al. 2003). In addition to this problem, PLSA is not a generative model of documents, as there is no clear way of assigning probability to a document outside of the training set. Blei et al. (Blei et al. 2003) identified the above issues with the PLSA model and proposed the new model of LDA. Girolami and Kaban (Girolami and Kaban 2003) have shown that when the LDA model is solved using *maximum a posteriori estimator* with a uniform Dirichlet prior, it results in the PLSA model.

The proposed solution is to take advantage of the assumption of the *bag-of-words* model in probabilistic models of text documents. The *bag-of-words* assumption is equivalent to an

[5] See the next section (section 2.3.4) for discussions of the KL divergence.

exchangeability assumption of the random variables involved. The *exchangeability* of random variables means that the joint distribution of the variables (words and topics in our case) is the same irrespective of the order of their appearance. A representation theorem due to de Finetti says that any collection of exchangeable random variables can have their joint distribution given by a mixture distribution. The components are the joint distribution of the random variables given the model parameter and the mixing proportions are distributions on parameters of the joint distribution which serve as priors. de Finetti's theorem can be written for an infinite mixture model as follows for three random variables x, y, and z

$$p(x, y, z) = \int p(\theta)p(x, y, z|\theta)\, d\,\theta \tag{2.17}$$

where θ is the parameter of the distribution $p(x, y, z)$ and $p(\theta)$ is a prior.

Applying de Finetti's theorem on the PLSA model gives rise to LDA. The graphical model of LDA is shown below (figure 2.3)

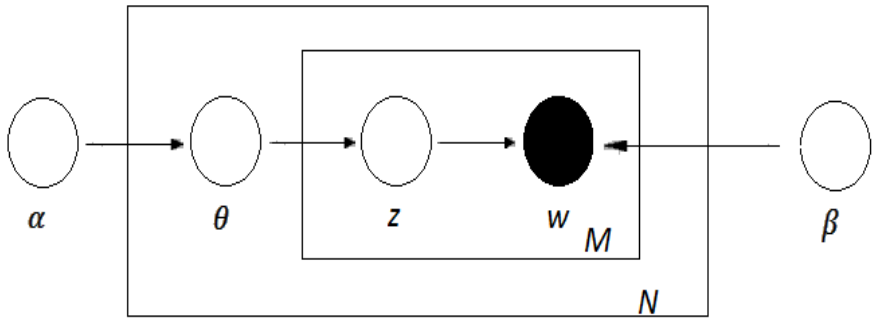

Figure 2.3: Graphical model of LDA.

(Blei et al. 2003) choose the Dirichlet prior for computational simplicity during inference and parameter estimation of the model. α is parameter of the Dirichlet prior. $\theta = (\theta_1, \ldots, \theta_k)$ is a document label that gives the values $p(z)$ for each topic z of a given document. θ corresponds to the d in figure 2.2 of the PLSA model. So, d is also a document label. The Dirichlet prior for k number of topics is given by

$$p(\theta|\alpha) = \frac{\Gamma(\sum_z \alpha_z)}{\prod_z \Gamma(\alpha_z)} \theta_1^{\alpha_1 - 1} \dots \theta_z^{\alpha_z - 1} \tag{2.18}$$

where $\Gamma(x)$ is the Gamma function. The graphical model of figure 2.3 results in the following generative process for LDA:

- Choose $\theta \sim Dir(\alpha)$
- For each of the M words, w_m
    - Choose a topic $z_m \sim Multinomial(\theta)$
    - Choose a word w_m from $p(w_n|z_m, \beta)$

The joint distribution of the model for a topic mixture θ, a set of k topics $\boldsymbol{z}$, and a set of M words $\boldsymbol{w}$ is given by

$$p(\theta, \boldsymbol{z}, \boldsymbol{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{m=1}^{M} p(z_n|\theta) p(w_n|z_n, \beta) \tag{2.19}$$

A number of inference techniques such as variational inference (Blei et al. 2003) and Gibbs sampling (Griffiths and Steyvers 2004) can then be applied to determine the various distributions of the model.

Unlike PLSA, folding-in in an LDA model can proceed in the same manner as the procedure used for finding the topic distributions of the documents of the original collection. The reason for this is that unlike PLSA, LDA is a generative model for documents.

We conclude the discussion on probabilistic topic models by giving similarity measures between probability distributions that are representations of words and documents (Steyvers and Griffiths 2007). Two popular measures are the Kullback Leibler (KL) divergence and the Jensen-Shannon (JS) divergence. The KL divergence between two distributions p and q is given by

$$D(p,q) = \sum_{j=1}^{T} p_j \log_2 \frac{p_j}{q_j} \quad (2.20)$$

The symmetric KL divergence is given by

$$KL(p,q) = \frac{1}{2}[D(p,q) + D(q,p)] \quad (2.21)$$

The symmetrized JS divergence is given by

$$JS(p,q) = \frac{1}{2}[D(p,(p+q)/2) + D(q,(p+q)/2)] \quad (2.22)$$

Another similarity measure that can be used is cosine similarity.

2.4. Approaches to Text Summarization

This section reviews the literature for the various approaches that are used for text summarization. This work closely follows the classification scheme adopted by Karen Jones (Jones 2007). She classifies extractive summarization approaches into four major categories: basic statistical approaches, enriched statistical approaches (lexical units and features), enriched statistical approaches (structures), and machine learning approaches. Machine learning approaches use techniques such as latent variable models and support vector machines (SVM), both in unsupervised and supervised settings. Graph-based approaches, which fall under *enriched statistical approaches (structures),* are discussed separately for emphasis. We can find concept representations in all but the *basic statistical approaches* which use surface level features to extract important sentences.

2.4.1. Basic Statistical Approaches

Basic statistical approaches are among the simplest approaches. The summarization algorithm might use a number of statistical features to measure the importance of sentences for inclusion in

the summary such as word frequency, position of a sentence, sentence length, presence of title words and cue words/phrases.

Luhn (Luhn 1958) presented the first automatic text summarization system. The statistical feature used is word occurrence frequency in the document. The feature is used to derive high frequency content words (keywords). He argues that both low frequency and high frequency terms are less helpful than mid-range frequency terms. The high frequency words can be identified either through the use of stop-list (list of stop-words) or a cut-off frequency. Sentences are scored based on the number of significant words (keywords) and the total number of words they contain.

The work by (Edmundson 1969) used three additional features in addition to Luhn's keyword approach: cue words, title words, and sentence location. Each sentence is given a score for each of these features and the individual scores are combined by a linear weighting mechanism to give an overall score for the sentence. Except for the score of a sentence by its sentence location, all the other scores of a sentence are the sum of the scores of its constituent words. The method using cue words use a dictionary of three types of words: bonus words, that contribute positive valued scores; stigma words, that contribute negative valued scores; and null words that contribute zero to the score. The title method is used based on the hypothesis that an author gives a title to the document in such a way that it describes the subject matter of the document. The sentence location feature is based on the hypothesis that topic sentences tend to occur in certain locations of the document, depending upon the type of document considered.

The word frequency approach of (Luhn 1958; Edmundson 1969) was later extended in many summarization systems by the use of the vector space model (Carbonell and Goldstein 1998; Gong and Liu 2001; Erkan and Radev 2004). These summarization systems represent passages (sentences) in the vector space and compute similarity scores via cosine similarity. Gong and Liu (Gong and Liu 2001) proposed a single-document summarization system based on this approach. Each sentence of a document is represented in the vector space. Then the cosine similarity of each sentence (relevance measure) is calculated with that of the whole document. A sentence is selected that has the highest relevance score and added to the summary. This sentence is

removed from the candidate sentence set and all the terms contained in it are removed from the original document. The steps of relevance measure, sentence selection and term elimination are repeated for the remaining sentences until the summary length is reached.

2.4.2. Enriched Statistical Approaches: Lexical Units and Features

These categories of approaches are characterized by the use of linguistic features to score sentences. Some of the methods include the use of recurrent n-grams (multiword descriptions), corpus-based statistics, part-of-speech tagging, semantic role labeling, named entity recognition, sentence parsing to identify grammatical make up of a sentence and use of lexical resources like WordNet (Miller 1995) and HowNet (Dong and Dong 2003). It is not uncommon for the summarization systems to use more than one of the methods cited above.

Filatova and Hatzivassiloglou (Filatova and Hatzivassiloglou 2004) proposed event-based summarization algorithm which was tested on DUC 2001 multi-document summarization task. An event can be broadly defined as something that happens at a given place and time. The summarization algorithm should thus be able to automatically detect events. The event detection algorithm is reported in their previous work (Filatova and Hatzivassiloglou 2003). Proper nouns, locations (usually proper nouns), time phrases and cardinal numbers were identified to be strong indicators of event regions. A named entity recognizer is used to extract event region indicators. The connector (in-between) words between named entities are also considered. They connect pair of named entities, called relations. The appropriate connectors were filtered out using a statistical parser to obtain parts of speech information and WordNet. Co-occurrence frequency and similarity of named entities and connectors is used to give score to relations and connectors. An atomic event is defined as a triplet of two named entities and a connector. Each atomic event has a score based on its relation and connector. These atomic events constitute concepts extracted from the text. The authors propose to select those sentences that together account for the majority of the concepts in the documents to be summarized. The authors make use of a greedy algorithm to avoid redundancy in their summary.

One of the problems faced by algorithms that use basic statistical approaches is the lack of identification of which parts of the text are related and in what way. They fail to identify cohesion in the text. Textual cohesion can be captured by a number of methods such as lexical chains and anaphoric chains. A lexical chain is a collection of semantically related terms while an anaphoric chain is one which is built from words or phrases that refer back to some previously expressed word, phrase or meaning. Creating representation of the document that gives information about semantically (conceptually) related terms contributes to identifying the main points of the document as illustrated by the works of (Barzilay and Elhadad 1997; Silber and McCoy 2002) that use lexical chains and (Bergler et al. 2003) that make use of anaphoric chains. The summarization system of (Barzilay and Elhadad 1997) uses WordNet, a part-of-speech tagger and a shallow parser to extract lexical chains. Lexical chains are scored based on the number of occurrences of members of the chain and the number of distinct occurrences. Sentences that contain high scoring chains are selected for the summary.

2.4.3. Enriched Statistical Approaches: Structures

Many summarization systems make use of document structures to create summaries. This increases the applicability of these algorithms when compared to knowledge intensive algorithms that are domain dependent and when compared to basic statistical approaches which tend to produce incoherent summaries (Ono et al. 1994).

Boguraev and Kennedy (Boguraev and Kennedy 1997) make use of anaphoric chains to identify the most salient (representative) phrasal units (noun phrases/nominal expressions) as well as expressions associated with them (relations) to generate topic stamps that capture the core content of the document. The authors' contribution in this paper is the generation of topic stamps, which are simple phrases which do not form a readable summary when put together. The authors' leave the subsequent use of the resulting topic stamps to generate a readable and concise summary as future work. Their summarization system has five stages. The first stage, discourse segmentation, applies a segmentation algorithm to the whole text to derive text segments that talk about the same topic. A segment usually consists of a number of related paragraphs. The second stage, phrasal analysis, extracts nominal expressions and relations from each segment.

Each expression serves as a candidate topic stamp. The third stage, local salience extraction, uses anaphora resolution to select the salient phrases/relations from each segment. Phrases/relations found at the third stage are too many for consumption of the user and hence a further reduction step is needed. This gives rise to the fourth stage, discourse salience extraction. The fourth stage calculates discourse salience which measures the importance of phrases/relations in each segment with respect to the whole document. This stage uses an extension of the algorithm used in the third stage. The final and fifth stage, topic stamp identification, involves selecting phrases/relations from each text segment according to their discourse salience.

A notable summarization approach that makes use of document structure is a Rhetorical Structure Theory (RST) based summarization system. RST is a method of describing the structure of texts in terms of the relationships that exists between them (Mann and Thompson 1988). Various types of relationships might exist between textual units/spans (clauses, sentences or paragraphs) in a document such as elaboration, background information, evidence, justification, restatement, and summary. Marcu (Marcu 1997a, Marcu 1997b) makes use of RST theory to build a summarization system. He defines two types of relationships between text spans. Paratactic relations are relationships between spans of equal importance. Hypotactic relations are relations between spans called nuclei (which are considered important by the author) to textual units called satellites (which are considered less important because they have a supporting role to the nuclei). Following this, the structure of the text represented by RST provides a binary tree structure. The rhetorical parser he uses considers clauses as text units. A text unit may reside at more than one node in the tree, possibly sharing a node with another textual unit. A given node in a tree may have the role of either a nuclei or a satellite. Satellite instances of a text unit are given zero scores. Nuclei nodes that are leaves are given a score of one. The score given to nuclei nodes is incremented by one as we move to the root. Hence, the root of the tree is given a score of n, where n is the depth of the tree. To obtain the final score of a text unit, we take its highest score. Summary is then generated by picking nuclei nodes that have the highest scores.

2.4.4. Graph-based Approaches

Graph-based ranking algorithms such as HITS (Kleinberg 1999) and PageRank (Brin and Page 1998) have been successfully applied in different domains such as citation analysis and analysis of link-structure of the World Wide Web. Variants of these ranking algorithms have been adapted to text summarization tasks in the form of unsupervised methods (Mihalcea and Tarau 2004a; Mihalcea and Tarau 2004b; Erkan and Radev 2004). Graph-based algorithms use textual units (such as sentences or words) to form nodes of the graph. Nodes share a link based on a similarity measure such as the number of common words or cosine similarity between sentences. These connections, called edges, can be of various forms: directed/undirected, weighted/un-weighted, and retaining all or some of the edges by applying a threshold value to their weights. When a link is created from one node to another, it is regarded as a "recommendation" or "voting", which is taken into account when calculating salience or importance score. The importance of a vote is determined by the importance of the voter with respect to the whole graph. All graph-based ranking algorithms arrive at a final score of each node by a recursive analysis of a vote given to a node and the importance of the voter, making use of the whole graph structure.

Mihalcea and Tarau (Mihalcea and Tarau 2004a) apply variants of the HITS and PageRank algorithms to create single- and multi-document summarization systems. Similarity between the nodes (sentences) is established based on the number of common tokens. After the graph-ranking algorithms are run, top ranking sentences are selected for inclusion in a summary. Melese (Melese 2009) applies a combination of graph-based and LSA-based algorithms, while (Bhandari et al. 2008) uses PLSA to create a graph-ranking algorithm that is a generalization of the HITS algorithm (see next section for a detailed review of these two works).

The graph-based methods discussed here measure similarity between sentences based on only word overlap. This means that these approaches use only surface level statistical techniques to discover salient sentences. However, the recursive analysis of importance of a node within a graph compensates for this "surface level" approach, as it has shown to produce competitive results with respect to systems that use relatively advanced summarization approaches (Mihalcea

and Tarau 2004a; Erkan and Radev 2004). This means that making use of graph-ranking algorithms on simple co-occurrence statistics produces language and domain independent summarization systems. This same argument applies to summarization systems based on latent variable models, though the compensating factor here is dimensionality reduction.

2.4.5. Summarization Systems based on Latent Variable Models

Text summarization systems based on latent variable models have gained significant momentum recently. Central to all systems is the decomposition of documents and words into topics that are smaller in number when compared to the number of sentences in the documents. After the topics of the documents have been identified, a variety of ranking algorithms can be applied to extract the most informative sentences. It is necessary to review works based on a variety of latent variable models (LSA, PLSA and LDA) because a ranking algorithm used in one model could be adapted for use in another in the hope of getting increased system performance and robustness to various document styles. We start with approaches based on LSA, then PLSA and finally LDA.

Gong and Liu (Gong and Liu 2001) present a fairly straightforward LSA based summarization system. After creating term by sentence matrix of a single document to be summarized, they use SVD to decompose the matrix. They select the sentence which has the largest index value with the k^{th} right singular vector, and include it in the summary. Here, k is set to 1 initially, which corresponds to the right singular vector with the largest eigenvalue. The operation is continued by incrementing k. When k reaches a predefined value (i.e., length of the summary measured in terms of number of sentences), the operation stops. The authors proposed this solution after presenting the argument that a good generic text summary of a document consists of the main topics of the document with minimum redundancy. Thus, by taking one sentence from each right singular vector, they reach their objective of selecting the representative sentences of the document. Their approach however has the drawback of possibly selecting sentences from topics that may not be important with respect to the document as a whole. This stems from the fact that the index k increments towards right singular vectors that may have smaller values of eigenvalue.

Steinberger (Steinberger 2007) improves upon Gong and Liu's approach by following a more elaborate approach. He makes use of work by (Ding 2005) that proves that the statistical significance of each LSA dimension is approximately the square of its singular value. This means that LSA dimensions with smaller values of the square of their singular values represent redundant and noisy information, i.e. unimportant topics. To exploit this result, Steinberger creates the matrix $\Sigma^2 . V^T$ and gets a new representation of the sentences different from the usual $\Sigma . V^T$. He then calculates the distance of each sentence from the origin of the resulting reduced vector space representation. This distance will be given as a score to each sentence. Sentences with the highest score are then selected for inclusion in the summary. This enables the algorithm to choose sentences with greatest combined weight across all topics, resulting in the possible inclusion of more than one sentence about an important topic. This is in contrast with Gong and Liu's approach that always selects one sentence for each topic.

Steinberger (Steinberger et al. 2007) further extends his LSA based system by making use of anaphoric information. He used a system that automatically identifies anaphoric chains. The first method (the substitution method) he used this information is by substitution of the anaphoric expressions with the first element of their anaphoric chain. He then uses the resulting term by sentence matrix for further processing. The second method (the addition method) does not modify the original terms of the document. Instead, the notion of a term is generalized to include anaphoric chains. The representation of a sentence then specifies the occurrence of both a word and a discourse entity from a particular anaphoric chain. While the substitution method degrades the performance of the system slightly, the addition method significantly improves the original LSA based summarization algorithm.

Melese (Melese 2009) implemented an Amharic summarization system based on LSA and graph-based algorithms which were tested on news texts. He proposes two new algorithms that he called TopicLSA and LSAGraph. TopicLSA starts by constructing the term by sentence matrix A for analysis by SVD. Melese uses the title of the document just like a sentence in the term by sentence matrix. He points out that this enables a possibly accurate representation of the title of the document in the latent space than folding the title of the document into the latent space that is constructed from sentences of the document excluding the title of the document.

After SVD is applied to matrix A, dimensionality reduction followed to give a reduced representation of A given by $A' = U'\Sigma'V'^T$. Unlike previous works that use V'^T to rank sentences of the document, he proposes to use U' to extract important concepts of the document. He argues that the term by concept matrix U' has richer representation of the concepts of the document than the concept by sentence matrix V'^T. For each column of the matrix $U'\Sigma'$ the top m terms that have high index value in the column are selected and all the resulting terms are concatenated to form a topic vector P. The topic vector is then folded into the latent space of $\Sigma'V'^T$ for similarity comparison with the sentences of the document. Sentences of the document have three scores. The first one is the cosine similarity of a sentence with P in the latent space. The second one is the cosine similarity of the sentence with the title of the document in the latent space. The third score is $1/n$, where n denotes the position of the sentence from the beginning of the document. The three scores are then summed up to give the overall score for the sentence. The sentences with the highest score are then selected for inclusion in the summary. The second and third scores are calculated to account for the characteristic of news article. The second score accounts for the fact that headlines of news texts carry the gist of the document as a whole. The third scores accounts for the fact that news texts usually put the most important information near the beginning of the document and the information importance of sentences decreases as the document progresses towards the end.

The GraphLSA algorithm is a mixture of graph-based and LSA-based approaches. It uses sentences for constructing the nodes of the graph. Unlike previous works, except for the work by Yeh et al. 2005), that use keyword-based frequency vector to represent each sentence, it uses the representation of the sentences given by $\Sigma'V'^T$ that results after SVD of the term by sentence matrix A. The cosine similarity of the sentences in this latent space is used to construct the edges between the nodes. To take the title of the document into account during the graph ranking algorithm, Melese takes into account the similarity of the title with a sentence (node) when the score of the sentence is calculated, resulting in a topic-sensitive graph-based ranking algorithm (Otterbacher 2005). This helps to penalize those sentences that do not belong to the main topic of the document. He experimented with the PageRank and HITS graph ranking algorithms. The top ranking sentences are then extracted to create the summary.

Melese carried out the experimentation on 50 Amharic news texts. The extraction rates considered were 20% and 30%. TopicLSA outperformed the LSAGraph + PageRank approach in both extraction rates by F-measure. TopicLSA got the same score of 0.47 in F-measure as LSAGraph + HITS in the 30% extraction rate task, while outperforming LSAGraph + HITS in the 20% extraction rate task. However, in the 20% extraction rate task, LSAGraph + PageRank gets a score of 0.38 compared to the 0.35 score of LSAGraph + HITS. Perhaps not surprisingly, TopicLSA – Genre, i.e., TopicLSA without taking into account the second and third scores described above (document genre specific scores), is outperformed by the same algorithm that takes into account genre specific scores. Thus, it is fair to compare TopicLSA – Genre with other LSA based summarization systems that do not take document genre information into account when ranking sentences. TopicLSA – Genre outperforms Gong and Liu's and Steinberger's method. The TopicLSA – Genre algorithm gets an F-score of 0.28 for the 20% extraction rate while it achieves 0.40 for the 30% extraction rate. However, the baseline systems for LSAGraph that do not use LSA and document genre information, PageRank and HITS, have the following F-scores: PageRank scores 0.37 and 0.44 for 20% and 30% extraction rates respectively and HITS scores 0.35 and 0.41 for 20% and 30% extraction rates respectively. This shows that the baseline graph-based algorithms perform significantly better than TopicLSA – Genere. But as discussed above, TopicLSA outperforms the LSAGraph algorithms. This indicates that TopicLSA makes efficient use of document genre information as compared to PageRank and HITS. This means that either the way the LSAGraph algorithms make use of document genre information needs to be improved or the graph-based algorithms are more robust than TopicLSA – Genre algorithm for different types of documents.

Bhandari et al. (Bhandari et al. 2008) present a summarization system using PLSA for the creation of extractive generic summary of single documents. They compare their works with the increasingly popular graph-based method for sentence ranking, HITS and LexPageRank, and LSA based algorithms, (Gong and Liu 2001). They take note of previous works (Cohn and Chang 2001; Richardson and Domingos 2002) to point out that these graph-ranking algorithms suffer from the topic drift problem. While the algorithms identify a community of nodes that have the strongest influence on the whole graph as a whole, they fail to recognize the contribution of other community of nodes that may have a slightly smaller yet important

influence on the graph. Since a generic summary of a document should cover the main topics of a document which are essentially different from one another in concept, the graph-based algorithms might produce summary sentences that are not representative of the document as a whole. LSA can be used to solve this problem but it has incorrect assumptions on the nature of discrete data such as text documents. This might make LSA based summarization systems ineffective.

The authors propose four different algorithms using PLSA: PROC 1, PROC 2, PROC 3 and PROC4. PROC 1 and 3 make use of the topics from the reduced probabilistic latent space given by PLSA. PROC 2 and 4 are graph-based methods obtained using PLSA. PROC 1 selects topic z with the highest probability $p(z)$ and makes use of $p(s|z)$ to pick the top ranking sentences with respect to this topic until the summary length is reached. PROC 3 makes use of the fact that PLSA divides the document into a number of topics. The algorithm creates score for sentences so that sentences which have good influence ranging over several topics score better. The sentence score is given by $p(s) = \sum_z p(s|z)p(z)$. PROC 2 is the same as PROC 1 except for the fact that it applies PLSA to the sentence-similarity matrix that results from the original term by sentence matrix. Cosine similarity is used to construct the similarity matrix. After the construction of the sentence-similarity matrix, the remaining procedure is the same as PROC 1. PROC 4 is essentially the same as PROC 3 except for the fact that it uses the matrix created in PROC 2 instead of the one created in PROR 1. PROC 2 and 4 are essentially the same as PHITS (probabilistic HITS) that was proposed by (Cohn and Chang 2001). HITS can be interpreted as applying SVD to a graph-representation (in this case the sentence-similarity matrix) and take only the communities associated with the most dominant component. PHITS makes use of PLSA instead of SVD to come up with a more accurate representation of the dominant communities of nodes in the graph (Cohn and Chang 2001).

Their experiment is carried out on DUC 2002 corpus. They experimented with various numbers of topics. PROC 3 got the best results on all of the ROUGE scores considered (ROUGE-L, 1, 2 and SU-4) using two number of topics only. The LSA based system gave the least scores. LexPageRank and HITS gave marginally better scores than PROC1, while they gave significantly better scores than their probabilistic versions PROC 2 and 4. Common sense logic

indicates that at least HITS should have performed less than PROC 2 and 4 as they are its exact probabilistic analogues. This suggests that perhaps PROC 2 and 3 do not make effective use of the resulting probabilistic graph-based representations.

Query-focused multi-document summarization with PLSA is investigated by Henning (Hennig 2009). Sentences are represented as usual by training the PLSA model from a multi-document set to be summarized. Query (document cluster narrative that expresses a user's information need) that is given as part of the summarization task is folded into the trained model. The cluster title, document titles, document term vectors and cluster centroid vector are also folded into the reduced latent space. Once we have a representation of the sentences and the five "queries" described above in the latent topic space, a number of sentence level features are calculated. Each sentence is given a score against one of the five "queries" by one of three similarity measures considered: KL divergence, JS divergence and the cosine similarity. All the individual scores for a sentence are then combined linearly into an overall score. The highest scoring sentences are selected for inclusion in the summary. An approach similar to maximum marginal relevance (MMR) (Carbonell and Goldstein 1998) is used to penalize candidate sentences based on their similarity to the partial summary. PLSA model that is trained with the EM algorithm with a random initialization converges on a local maximum of the likelihood of the observed data. Different initializations will result in different locally optimal models. To overcome this issue, following the work by (Brants et al. 2002), the authors compute the sentence level features for five different models and average the results. The authors used DUC 2006 data for training the feature weights used to combine the sentence level scores linearly. They used DUC 2007 data for testing their system. Their model got very competitive performance results when compared to the participating systems of both DUC 2006 and 2007 participating systems. The JS similarity measure resulted in the best ROUGE scores and gave the most stable scores when the number of latent topics was varied.

Arora and Ravindran (Arora and Ravindran 2008) apply LDA for extractive multi-document summarization. Their summarization algorithm starts with a term by document matrix and not a term by sentence matrix as is usually the case. After the LDA model is trained, the different probability distributions result: $p(z|d)$ and $p(w|z)$. They use three different algorithms to

calculate the probability of a sentence given a topic, $p(s|z)$: simple inference based, partial generative and full generative. All of the algorithms assume that a sentence of a document belongs to only one topic. They differ on the assumptions they make on how the topic of a sentence is generated by the words that it contains and the document that it belongs to. Because $p(s|z)$ contains the product of $p(w|z)$ for all the words that it consists of, longer sentences will be penalized as $p(w|z) < 1$. This is a good observation by the authors since in many summarization algorithms the issue is that longer sentences are favored over shorter ones. The authors come up with modified versions of the above algorithms to resolve this issue. For the summary generation, the probability of a topic $p(z)$ is calculated and the following iterative procedure is used until the summary length is reached: do a multinomial sampling on $p(z)$, choose the sentence with the highest value of $p(s|z)$, and if it is already included in the summary pick the sentence with the next highest probability for this topic. They evaluated their algorithms on DUC 2002 data and got significant improvement in ROUGE-1 recall measures compared to the top ranking systems. The partial generative algorithm gets the best score followed by the simple inference based algorithm. We cannot make use of the work of these authors since they use a term by document matrix for input to the LDA model. This means that extra work needs to be done to calculate $p(s|z)$, which would not be the case if a term by sentence matrix is used to train the LDA model.

2.4.6. Amharic Text Summarization Systems

This section discusses attempts to create an Amharic text summarization system. The work by Melese (Melese 2009) is discussed in section 2.4.5 together with other summarization systems based on latent variable models. Common to all systems discussed here is the use of surface level statistical features to extract important concepts of the document.

The first Amharic summarization system was developed by Kamil Nuru (Kamil 2004), a single document summarizer. He used surface level statistical features to assign weights to sentences. The highest scoring sentences were extracted to form the summary. Seven features are used: title words, cue phrases, first sentence of the document (header), words in a header, first sentence of a paragraph, paragraph end sentences and high frequency words (keywords). Each feature has an

associated weight, obtained from training with manual summaries of four news articles, and the weights are combined linearly to produce an overall score for a sentence. As part of a learning phase, the stop-words and cue phrases lists of the system can be updated by a user that is generating a summary. The system is tested with five news articles at 38.5% extraction rate. The average sentence length of the five news articles is 17.4 sentences, the lowest length being 13 sentences and the highest 23. The system achieves a precision of 0.704, recall of 0.58 and an F-measure of 0.636. The most dominant features in the experiment are title words and keywords. For the feature calculated using keywords, the percentage of keywords used for scoring and the weight given to this feature gave significant variations in system performance. Kamil needed to use 33% of the keywords identified to achieve the best result. He points out that because no stemmer is used for preprocessing the document, the frequency assigned to keywords is not accurate and hence less percentage of keywords could have been used if a stemmer was included. He also points out that the use of exhaustive stop-words could improve system performance.

Teferi (Teferi 2005) uses Naïve Bayes classifier (using the machine learning tool WEKA (Hall et al. 2009)) to create a single document summarization system. 480 news articles are used for experimentation, where 20 of them are used for testing and the rest for training. The test set has an average sentence number of 9.2 sentences. The 480 articles have an average of at most 6.7 sentences. Each document has a corresponding manual summary created by three different summarizers. The features used for training are presence of title words, location feature (first, middle, and last) and keyword (frequent words). Summarization is performed at 30% extraction rate. The system has a precision of 0.8, recall of 0.833 and an F-measure of 0.816. These high values could be attributed to the very short news articles considered for testing purposes. For the keyword feature, the use of the top three keywords has been shown to be sufficient. This seems to be a reasonable quantity due to the use of very short articles for the experimentation. Teferi reports that the use of a single feature for summarization gives a very poor result while the use of all features results in the best system performance.

Helen (Helen 2006) produced a single document summarizer for legal judgments. She uses two statistical features to score sentences: cue phrases and sentence location. Documents to be summarized are manually segmented into five thematic areas: introduction, reason, fact, judicial

analysis and decisions. A summary is generated for each segment at a given extraction rate and combined to give a single summary. Ten documents were used for testing. The summarization method is unsupervised as no training is performed. Sentences with the highest weight are selected at 20% and 10% compression rates from each segment and are merged together to serve as a single summary for the document. The system has the following performance: at 20% extraction rate, precision = 0.339, recall = 0.576, F-measure = 0.427 and at 10% extraction rate, precision = 0.392, recall = 0.506 and F-measure = 0.441. These performance values might decrease if the segmentation was made automatically.

We conclude review of Amharic text summarization systems by discussing three additional works. All of these works extend the previous works by the use of additional surface level statistical features. Daniel (Daniel 2006) uses two new additional statistical features that were not explored by previous works to create a single document summarizer. These are frequency of numeric terms in a sentence and the total sum of term frequency of every word in a sentence. Other features used are position of a sentence and number of title words in a sentence. The system gives the best performance when only the position feature is used. Abraham (Abraham 2007) developed a multi-document summarizer using surface level statistical features. He used two additional features not explored by previous works: context-sensitive frequency-based score (CSF), which was introduced by (Nenkova et al. 2006), and centroid score. The other features are number of title words in a sentence and position of a sentence in a text. The CSF feature is calculated by taking the average of the probabilities of content words (keywords) in each sentence. The probability of a content word is its relative frequency in the documents to be summarized. Context sensitivity is incorporated in the CSF score to avoid redundancy in the resulting summary. It is achieved by decreasing the probability of a word that is included in the summary to a very small value close to zero. The centroid score for a sentence is computed as a cosine similarity between a sentence and the document cluster centroid, where the cluster centroid is defined as the average of the *tf-idf* values of the sentences in the vector space model representation. Sentences are scored by a combination of the four features and by the individual feature values. The highest scoring sentences are selected for inclusion in the summary. The best system performance is obtained when the individual features of CSF feature and number of title words in a sentence are used. Another multi-document summarization system is that of (Winta

2009) which is based on WEKA. She explored three additional features that are not present in the work of (Teferi 2005). These are the sentence length cut-off, *tf-idf* value and the centroid score.

2.5. Research Problem

This chapter has explored in detail the literature concerning the task of automatic text summarization. The review has shown that representing the content of a text based on the language and domain-independent methods of topic modeling represents a promising approach to text summarization. This is especially true for local languages such as Amharic which lack extensive linguistic resources that can be used to identify important points/concepts of a document.

In this thesis, we further explore the use of topic modeling for text summarization. In particular, we try to build on the work of Melese (Melese 2009), the TopicLSA algorithm. Melese (Melese 2009) has shown that previous LSA-based approaches for the summarization of single-documents can be improved by taking advantage of the term by concept matrix that results from applying LSA to the document. As discussed in section 2.4.5, TopicLSA first selects some important words of the document and uses them to rank sentences. Thus, his approach can be called a *keyword approach*. Except for TopicLSA, previous approaches for summarization of documents using topic modeling have used only the sentence features that result from the topic modeling procedure used. We call them *sentence-based approaches*. In contrast to the sentence-based approaches, we plan to extend the work of (Melese 2009) by adopting the keyword approach for the summarization task. The main argument to consider the keyword approach or in general using words in our analysis and not just sentence scores is that, to decide relevance of a sentence for the summary, focusing on its smaller units (words) could give us a better insight into its importance. While the sentence features that result from topic modeling give us overall judgment of the sentence, only a few selected constituent words could be effective to decide relevance of sentences for inclusion in the summary. By following the keyword approach, we plan to investigate the contents of a sentence in a greater detail at the word level than at the

sentence level. Experiments by (Melese 2009) support this argument and have shown promising results. Therefore, we intend to further explore the use of keywords for sentence ranking.

Accordingly, we formulate our research problem in the following manner:

1. Proposing new methods for language and domain-independent concept-based single-document Amharic text summarization using keyword approach via topic modeling.
2. Once we design the appropriate sentence ranking algorithms, we will ask what type of relationships exists between the keyword approaches and sentence-based approaches. This helps us to pinpoint where the strong or weak points of the keyword-approaches lie with respect to the sentence-based approaches.
3. We will carry out evaluation of the different approaches we propose so as to identify the best performing approaches.

Chapter 3. Keyword Approach for Single-Document Amharic Text Summarization

This chapter gives a detailed treatment of the algorithms we proposed using the keyword approach to summarize single-document Amharic texts. We adapt the work of (Melese 2009) by first identifying points of improvement in his work. This is covered in section 3.1. In section 3.2, we try to select which of the topic modeling approaches (LSA, PLSA, and LDA) reviewed in chapter 2 could be used in our summarization algorithms. Section 3.3 presents the architecture of our summarization system; in total we propose six algorithms that explore the keyword approach for text summarization. Section 3.4, 3.5 and 3.6 discuss the component of the proposed summarization algorithms. Section 3.4 discusses the text pre-processing component. In section 3.5, the topic identification process using topic modeling is discussed. The proposed sentence ranking algorithms are detailed in section 3.6. Section 3.7 gives the summary to the chapter.

3.1. TopicLSA as a Keyword Approach

We start by giving mathematical representation of the ranking algorithm used by TopicLSA. In these discussions, we will not focus on the score pertaining to document genre as our main interest is in TopicLSA's use of keywords to rank sentences. Let us write the score given to a sentence S_j by TopicLSA – Genre using the discussions of section 2.4.5 and using the notations of equation 2.8. Let Q be the topic vector, $Q = [t_1\ t_2\ \dots t_i \dots\ t_m]^T$, where t_i represents the i^{th} term count. We fold Q into the latent semantic space to produce q given by equation 3.1 below

$$q = Q^T U' = \left[\sum_{i=1}^{m} t_i u_{i,1} \quad \dots \quad \sum_{i=1}^{m} t_i u_{i,c'}\right] \tag{3.1}$$

S_j is represented by the j^{th} row of $V'\Sigma'^T = A'^T U'$. This shows that the representation of sentence S_j in $V'\Sigma'^T$ can be obtained in a similar fashion as that of Q by using the j^{th} column of A' as the term counts. These term counts obtained from the reduced dimension representation of A' are more reliable than the original term counts of A. If we define $\cos\theta_{q,S_j}$ as the angle

between q and S_j, the score given to sentence S_j by TopicLSA – Genre becomes (equation 3.2 below)

$$Score(S_j) = \cos\theta_{q,S_j} \tag{3.2}$$

As equation 3.2 shows, TopicLSA - Genre chooses a topic mixture/proportion that is deemed important (represented by angle of q) and compares it to the topic mixtures (angles) of the candidate sentences. Sentences that are closest to q are selected as summary.

The term selection procedure of TopicLSA - Genre is not tailored towards the algorithm used for sentence ranking. Either of the two algorithms needs to be modified so as to give a logically correct summarization algorithm. We will discuss the required modification to the term selection procedure.

The term selection procedure of TopicLSA – Genre selects the top ranking terms from each column of U' and concatenates them to form a topic vector. The selected terms from a single column, say *column 1*, represent a certain topic but these terms may not rank highly with respect to other topics/columns. Other topics are represented by selecting terms from other columns. The sentence ranking procedure of TopicLSA – Genre then folds the resulting topic vector into $V'\Sigma'^{T}$ and compares it with the sentences of the document. This folding process is not theoretically sound with respect to the terms chosen previously. The folding process results in the representation of a word of the topic vector in the sense of all topics of the document rather than representing it in the sense of the specific topic from which it was selected during the term selection procedure. Let us describe this in some detail.

Assume t_1 of Q is selected from *column 1* of U'. This means that $u_{1,1}$ has a larger value when compared to terms that are not selected but other values $u_{1,j}$ may have low values when compared to other terms. It is these values that affect TopicLSA - Genre. As equation 3.1 shows q is represented by the values of $\sum_{i=1}^{m} t_i u_{i,j}$ with respect to each topic. Thus, each term of the topic vector contributes to the representation of q with respect to each topic via $t_i u_{i,j}$. In

particular, the contribution of $t_1$for *topic 1* of q is $t_1 u_{1,1}$. This value of $t_1 u_{1,1}$ is guaranteed to be high with respect to other terms. However, the contributions of t_1 to other topics of q, $t_1 u_{1,j}$, are not guaranteed to be high with respect to terms that have not been selected in Q because the term/keyword selection procedure did not consider the overall length of t_1's representation in U'. Terms like t_1 thus create q that tends to select sentences that are dominant in one topic but lack representation in other topics. This is equivalent to shifting the vectorial representation of q in the latent document space to favor certain topics only. This is not a good effect because we want to select sentences that score high with respect to all topics to create a better generic summary. The sentence ranking procedure is thus forced to create a score for a sentence by taking into account the importance of each word of the topic vector with respect to all topics rather than with respect to only the topics for which it was selected for.

As identified above, one way of alleviating the problem with TopicLSA – Genre is to select keywords by ranking them with respect to their length in $U'\Sigma'$. This helps avoid scoring a sentence with respect to a word that may rank higher with respect to a certain topic but may rank quite low with respect to other topics.

3.2. Choice of the Topic Model for Text Summarization

In chapter two, we discussed a number of topic modeling approaches: LSA, PLSA and LDA. We now choose one of them to extract topic-based features from our documents.

As discussed in section 2.3.2, LSA is a topic modeling approach that makes use of SVD to achieve dimensionality reduction. Underlying SVD is the assumption of Gaussian distribution of the words and documents. This Gaussian assumption may not suit discrete data like textual documents. The result is reduced performance as demonstrated by previous works such as (Hofmann 2001). As discussed in sections 2.3.3 and 2.3.4, probabilistic topic models such as PLSA (probabilistic latent semantic analysis) and LDA (latent Dirichlet allocation) do not suffer from this problem. This is the case because they rely on maximizing the likelihood function of the data and therefore do not fix the underlying probability distributions beforehand like LSA does. Since the aim of this thesis is to make progress towards a high quality summarizer, we

strive to use the best possible set of tools available. Thus we prefer the use of probabilistic topic models (PLSA or LDA) instead of LSA.

The next step is to decide which of the probabilistic topic models we reviewed to apply for the summarization task. Both PLSA and LDA are quite similar approaches. LDA differs from PLSA in that the topic distributions are assumed to have Dirichlet prior. This puts a limit on the number of parameters that are used to describe the resulting topic model to $km + k$, where k is the number of topics and we have an $m \times n$ term by document matrix (Blei et al. 2003). In case of PLSA, because documents do not have a generative model, the number of parameters becomes $km + kn$.

In case of information retrieval applications, this gives an advantage for LDA in terms of overfitting issues. Because LDA has less number of parameters, it is less likely to overfit when compared to PLSA (Blei et al. 2003). When we come to our task of text summarization of single documents the shortcoming of PLSA with respect to LDA, namely reduced predictive performance on new documents, is not present. This is because our trained probabilistic topic model is not used to predict topic distributions for future sentences as in the case of information retrieval applications. The purpose of the probabilistic model is just to understand the latent semantic property of the given or "training" document. In this regard, PLSA has an advantage over LDA. In the case of LDA, the number of parameters is smaller because we want to have generalization capacity to accommodate future documents. This may result in a less accurate identification of the topics of the document corresponding to each sentence. However, since PLSA has more number of parameters, each sentence is allowed to have its own specific topic distribution during parameter estimation. In this manner, we may "overfit" our topic model with accurate topic representations of the document by the use of PLSA with more number of parameters. Due to the above advantage of PLSA over LDA, we prefer to apply PLSA for our text summarization task.

3.3. Keyword Approach for Text Summarization using PLSA

The keyword approach for text summarization is based on the argument that using properties of sentences directly obtained from latent variable models such as $p(s|z)$ may not match the complex requirements of the text summarization process. Instead, we can switch to units such as words and phrases that have smaller granularity of statistical salience than whole sentences. One way of achieving this is to consider selected keywords based on some criteria and use them in the summarization process. Because the approach we follow is extractive summarization, the summarization process will consist of choosing sentences that best contain the selected keywords.

An architecture that shows the work flow of our approach to the development of Amharic text summarization system is shown in figure 3.1. The associated user interface can be found in Appendix G. Appendix H contains a sample generated summary using the JWS algorithm. The system has five modules: text pre-processing, topic identification, sentence ranking, sentence concatenation, and evaluation. Text pre-processing prepares the document in a format that is suitable for further analysis. At the topic identification stage, topic-based features are generated using PLSA. The sentence ranking module ranks sentences of the document and selects the top scoring sentences according to the extraction rate. The sentence concatenation component is responsible for putting together the resulting sentences to produce the document summary. Our architecture also contains an evaluation component whereby the resulting summary sentences from each algorithm are assessed for performance using precision/recall (see chapter 4 for details). Subsequent sections discuss the main sub-systems in detail (text pre-processing, topic identification and sentence ranking).

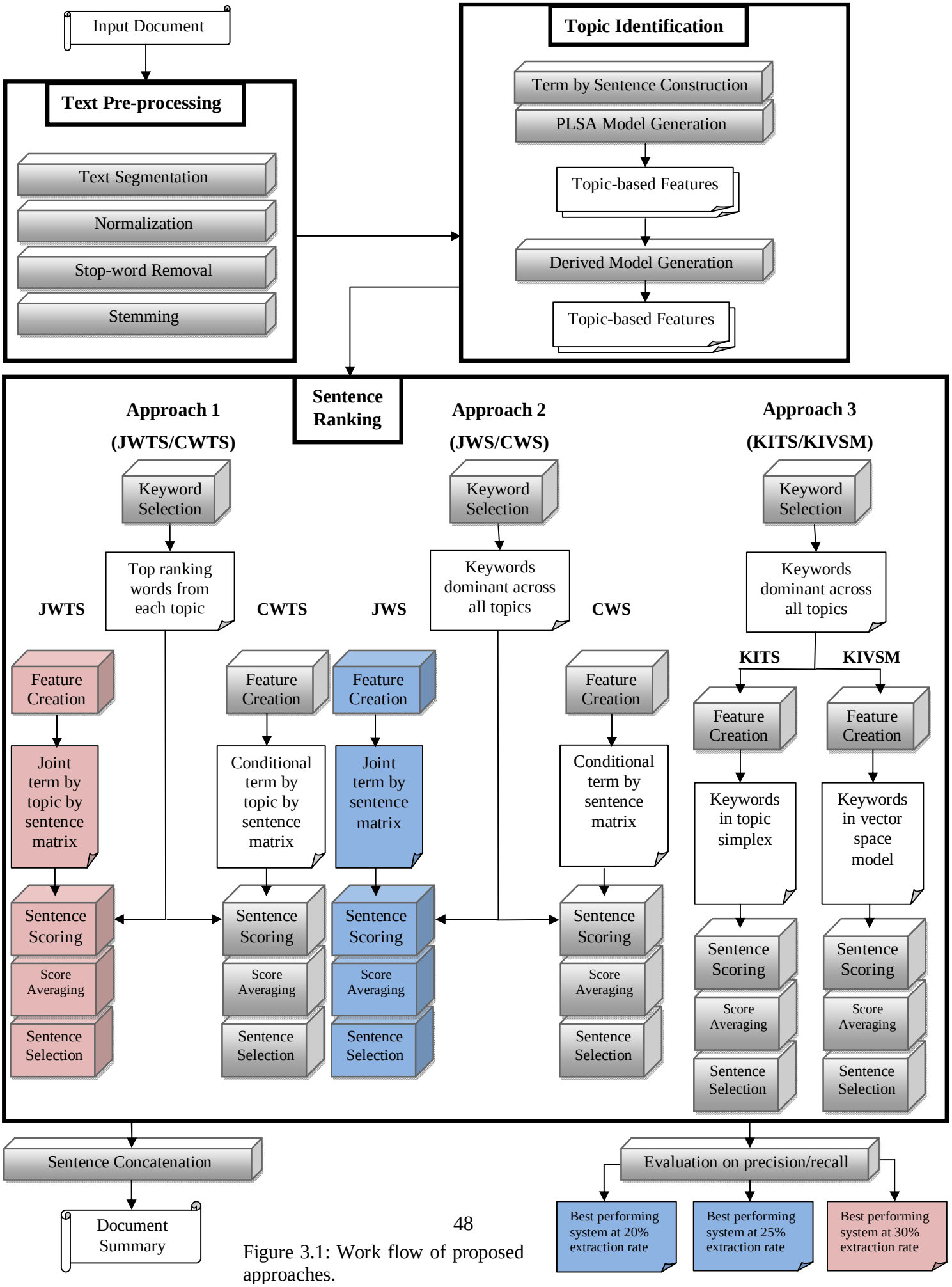

Figure 3.1: Work flow of proposed approaches.

3.4. Text Pre-processing

Text pre-processing is involved in preparing the text document into a format that is suitable for the text summarization process. Because our sentence ranking algorithms do not make use of the title of the sentence for ranking purposes, the titles are removed manually from the documents before the text pre-processing stage begins. The pre-processing stage consists of four steps that are executed sequentially in the given order: text segmentation, normalization, stop-word removal, and stemming. Text segmentation is the critical part of the pre-processing stage. Without it there is no way of identifying the words and sentences of the document for co-occurrence counts. Normalization, stop-word removal and stemming are optional and could be left out from the summarizer. Normalization and stemming help to create accurate co-occurrence counts by identifying words that represent the same or nearly identical meanings.

This work makes use of the pre-processing module adopted in (Melese 2009) for his Amharic text summarizer using LSA and graph-based ranking algorithms. Melese in turn used the pre-processing module from the work by (Tessema 2007) for Amharic search engine and adapted it for his text summarization work.

Text Segmentation

Text segmentation involves the identification of sentences and words of the document to be summarized. In the Amharic language, sentences are identified by the delimiter "።". Words are identified by means of spaces and a number of Amharic punctuation marks such as ፡, ፤, and !.

Normalization

Normalization is performed on the word tokens that result from text segmentation. In the Amharic language two types of normalization issues arise (Tessema 2007).

The first one is the identification and replacement of shorter form of a word that is written using forward slash "/" or the period ".". An example is the replacement of "ዶ.ር" by "ዶክተር". The list of short words used in this thesis are extracted from the work of (Tessema 2007). See Appendix B for the full list of short words and their expanded forms.

The second normalization issue is the identification and replacement of Amharic alphabets that have the same use and pronunciation but different representation. The replacement is made using a representative alphabet from a set of similar alphabets. For example, the word "wealth" can have many representations in Amharic: "ሀብት", "ሃብት", and "ኀብት". These three words differ only by their first characters: "ሀ", "ሃ", and "ኀ" have similar usages but different forms. They need to be converted to a single representative character such as "ሀ". Table 3.1 below shows some of the character replacements that are used in this thesis which are borrowed from the work of (Tessema 2007).

Table 3.1: Example character normalizations.

Character to be normalized							**Character after normalization**						
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ	ሀ	ሁ	ሂ	ሀ	ሄ	ህ	ሆ
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ሀ	ሁ	ሂ	ሀ	ሄ	ህ	ሆ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
ጐ							ጎ						
ኰ							ኮ						

In the table above, the alphabet "ሐ" normalizes to "ሀ", "ሑ" normalizes to "ሁ", and so on up to "ሖ" which normalizes to "ሆ".

Stop-word Removal

Stop-words are known to be useless in identifying the central points of a text. On the other hand, they may contribute significantly to the co-occurrence counts of the term by sentence matrix of a document to be summarized. This negatively influences the contribution of non-stop-words towards the identification of the central points of the text. As a result, it negatively affects the performance of the summarization algorithms. In order to avoid the effect of stop-words in the summarization algorithms, we remove stop-words from sentences of the document to be summarized. In this work, 153 stop-words are used, all taken from the work of (Tessema 2007) and (Winta 2009). See Appendix C for the list of stop-words.

Stemming

Very often, a word may appear in many different forms throughout a document. These different forms of the word essentially share identical meanings. Morphology is considered the main

source of the different forms of a word (Nega and Willett 2002). Morphology can be inflectional or derivational (Nega and Willett 2002). Inflectional variations of a word introduce changes in tense, number, case, and gender while keeping the same meaning of the word. On the other hand, derivational ones apply affixes to a word to result in the formation of new words.

If various forms of a word appear in a document to be summarized, it may have a degrading impact on the summarization process. By applying the process of stemming, different words that have the same stem can boost the representation of the concept underlying that stem in the term by sentence matrix. This helps to provide more data for the PLSA algorithm to work with because sentences that did not share the same term before stemming will now be connected by the stems of the words. For example, let us say these three words appear in a document in three different sentences; ቤት (house), ቤቶች (houses), and ቤታችን (our house). These words literally refer to the concept of a house. If stemming is not used, then all of them are treated as unique words in a document and appear different to the PLSA algorithm. However, if stemming is used, then all of the words are changed to ቤት (house) and hence the three sentences will be related by this word. This enables the PLSA model to use the resulting connection between the sentences to infer conceptual relations between other words contained in the three sentences.

This work makes use of the stemming algorithm adopted by (Tessema 2007) from the work of (Nega and Willett 2002). Nega and Willett (Nega and Willett 2002) handle both inflectional and derivational morphology, Tessema adopts only the inflectional one, arguing that stemming for derivational morphology may have a negative impact on the performance of the Amharic search engine. The stemming algorithm implemented by (Tessema 2007) removes those affixes (suffixes and prefixes) that form inflected forms of a word. He used 33 suffixes and 17 prefixes (see Appendix D for the list of affixes). The suffix removal involves two cases. If the suffix does not contain a vowel, then remove the suffix without making any further modifications to the remaining characters of the word. Otherwise the suffix is removed and the last character of the word is changed to “sadds” (ሳድስ), which is the six order of a character.

3.5. Topic Identification

The dimensionality reduction methods discussed in section 2.3 in general and PLSA in particular require the identification of co-occurrence counts of each unique word in a document to be summarized within each sentence. This is performed by the term by sentence matrix construction stage. The PLSA model generation stage is responsible for calculating the parameters of the asymmetric PLSA model. Finally, other parameters of the PLSA model will be generated by the derived model generation phase.

Term by Sentence Matrix Construction

The first stage of topic identification is term by sentence matrix construction. This stage makes the document to be represented in a way that the PLSA algorithm can process. As discussed in section 2.3.1, we create a vector space model (VSM) for a document. We represent the VSM by an $m \times n$ matrix where m is the number of unique words in the document and n is the number of sentences in the document.

This work makes use of the corresponding component of the Amharic text summarizer of (Melese 2009). Each row of the matrix represents a single unique word, while each column represents a sentence. The code by (Melese 2009) first sorts all the words alphabetically in ascending order and then associates each row of the matrix to the sorted words in that same order (for example, the first word in the sorted list will be represented by the first row). In the case of sentences, the sentence that appears first in the document is represented by the first column, the sentence that appears second in the document is represented by the second column, and so on up to the sentence that appears last in the document which is represented by the last column. An entry at the i^{th} row and j^{th} column of the matrix is filled with the count of the i^{th} word in the j^{th} sentence.

PLSA Model Generation

This stage is responsible for generating topic-based features using PLSA. For this step, we wrote Java code that implements the asymmetric model that was proposed by (Hofmann and Puzicha 1998; Hofmann 2001) (see section 2.3.3 for details). The algorithm implements the tempered

version of the EM (Expectation Maximization) process while generating the PLSA model. We use tempered EM to see if better summarization performance can be achieved. While (Hofmann and Puzicha 1998; Hofmann 2001) introduced tempered EM to avoid overfitting, we use it to avoid unfavorable local maxima during the EM procedure.

Because EM is an iterative maximization process, we have to decide when to stop the iterations. The function that we actually maximize is the predictive log-likelihood, $\mathcal{L}p$ (where $\mathcal{L}p = \sum_{d\epsilon D} \sum_{w\epsilon W} n(d,w)\ log\ p(w|d)$), rather than the joint log-likelihood. Maximizing using either is equivalent (Hofmann and Puzicha 1998). To decide when to stop, we calculate a value called convergence criterion (%). It is given by equation 3.3 below

$$convergence\ criterion\ (\%) = \frac{\mathcal{L}p(previous) - \mathcal{L}p(current)}{\mathcal{L}p(previous)} * 100 \qquad (3.3)$$

The iterative procedure stops when convergence criterion (%) is less than a fixed number (see section 4.3.1 for details on the "fixed number" used). In addition to checking the condition for convergence criterion (%) for convergence, we also introduce the requirement that a minimum of five iterations be made. This makes sure that the resulting PLSA model will not suffer from undertraining.

If we start with an $m \times n$ term by sentence matrix, then we arrive at the following topic-based features of the PLSA model:

- A $1 \times n$ normalized sentence length matrix $p(s)$.
- An $m \times k$ term by topic matrix, $p(w|z)$, where k is the number of topics of the PLSA model.
- A $k \times n$ topic by sentence matrix $p(z|s)$.

The parameter $p(s)$ is the normalized sentence length (number of non-stop-words contained in a sentence). For a fixed latent topic z, $p(w|z)$ represents a probability distribution that gives the degree to which words of the document represent that particular topic. For a fixed sentence s,

$p(z|s)$ represents a probability distribution that gives the degree to which a topic z is present in that sentence.

Derived Model Generation

The previous stage calculates basic parameters of the asymmetric PLSA model. However, the proposed algorithms also require the computations of other parameters of the PLSA model that are actually basic parameters of the symmetric model of PLSA. These parameters are $p(s|z)$ and $p(z)$. For a fixed value of z, $p(s|z)$ gives the degree to which a sentence of the document represents that particular topic. $p(z)$ indicates which topic is likely to be mentioned frequently in the sentences of the document. A higher value of $p(z)$ indicates that the corresponding topic is referred to more often than the other topics in the document.

The parameter $p(z)$ can be computed by marginalization of $p(z, s)$ (equation 3.4 below)

$$p(z) = \sum_{s} p(z, s) = \sum_{s} p(s)p(z|s) \tag{3.4}$$

The parameter $p(s|z)$ can be computed using the basic parameters of the asymmetric PLSA model, $p(z)$ and Bayes's rule as given by equation 3.5 below

$$p(s|z) = \frac{p(s)p(z|s)}{p(z)} \tag{3.5}$$

3.6. Sentence Ranking

In this section, we propose new methods of sentence ranking that make use of conceptual representations of both words and sentences as obtained from a PLSA model. Unlike TopicLSA whose keyword selection and sentence scoring algorithms are not in tune, the sentence ranking algorithms we propose here do not have this problem.

We explore the keyword approach for sentence ranking by proposing a total of six algorithms. All the algorithms will be implemented and compared against one another to select the algorithms with the best performance for the three extraction rates: small (20%), medium (25%) and large (30%) summaries. These six algorithms are divided into three categories: Approach 1, Approach 2, and Approach 3. Approach 1 consists of two algorithms: JWTS (joint word topic sentence) and CWTS (conditional word topic sentence). Approach 2 consists of two algorithms: JWS (joint word sentence) and CWS (conditional word sentence). Approach 3 also consists of two algorithms: KITS (keywords in topic simplex) and KIVSM (keywords in vector space model).

All of the six algorithms have five phases: keyword selection, feature creation, sentence scoring, score averaging, and sentence selection. During the keyword selection phase, words of the document that are judged to represent the most important points are selected. After this, a feature that helps characterize sentences of the document is created using the PLSA model. Then, the feature will be used to give scores to sentences. The above three steps are then repeated a number of times, $numModel$ times. This results in $numModel$ different scores for the sentences of the document. These scores are then averaged to give the final score to the sentences. This procedure of score averaging is used to counter act "poor" models that might result from unfavorable local maxima (Brants et al. 2002; Hennig 2009). Finally, the sentence selection stage uses the average score to select the top ranking sentences. In addition, the sentence selection stage always selects the first sentence of the document if the average score assigned to it does not make it one of the top ranking sentences. This is carried out to take advantage of the property of news articles (the documents we evaluate our summarization algorithms with) where the first few sentences of a document hold important information (Nenkova 2005).

We have considered two keyword selection strategies. In the first approach (followed by Approach 1), top ranking keywords are selected from each topic. In the second approach (followed by Approaches 2 and 3), top ranking keywords of the document are selected that are dominant across all topics.

In all cases, the sentence scoring phase attempts to estimate the presence of the selected keywords in each sentence. The basic idea is that, if a sentence contains more number of keywords than other sentences, then it will be selected for the summary. We have considered two ways of estimating the presence of keywords in sentences (Hofmann 1999b; Buntine et al., 2004; Steyvers and Griffiths 2006). In the first approach (followed by Approaches 1 and 2), sentences that maximize the joint/conditional probability of the keywords and/or topics given a sentence are selected. In the second approach (followed by Approach 3), the keywords are represented in the same manner as the sentences in a space (the topic simplex in the case of KITS and the VSM in the case of KIVSM) and sentences that are most similar to the keywords (based on a similarity measure such as cosine) are selected.

The score averaging and sentence selection phases are identical to all of the algorithms. For this reason, they are discussed last in sections 3.6.4 and 3.6.5 respectively. The detailed discussion of each algorithm, without the score averaging and sentence selection components is presented in the sections that follow.

3.6.1. Approach 1

Approach 1 consists of the algorithms JWTS (joint word topic sentence) and CWTS (conditional word topic sentence). It is characterized by having a keyword selection strategy where top ranking keywords are selected from each topic. It follows two different sentence scoring strategies. JWTS assigns as a score to a sentence the joint probability of the keywords, the topics from which each keyword is chosen from and the sentence. CWTS assigns as a score to a sentence the conditional probability of the keywords and the topics from which each keyword is chosen from given the sentence. Each algorithm is discussed in detail in the sections that follow.

3.6.1.1. JWTS Algorithm

Keyword Selection

A good summary usually has a wide coverage of the document's main topics (Gong and Liu 2001; Bhandari et al. 2008). We represent the main contents of the document by using keywords that best represent each of the main topics of the document. The keywords that best represent the

topics of a document can be inferred from the basic parameter of the asymmetric PLSA model, $p(w|z)$. To give a topic z a representation in the document summary, we select L number of words that have the highest value in the probability distribution $p(w|z)$. This is done for each topic that has been generated from the PLSA model. In this way, all topics of the document are represented by a selected set of keywords. The total number of keywords selected is thus $L \times K$, where K is the number of topics.

Feature Creation

This stage assigns a set of features to each sentence of the document. For a single sentence, $L \times K$ number of features is calculated. A feature corresponds to a single keyword and the topic from which it is chosen from. The feature basically measures the degree to which the sentence contains the keyword in the sense of the topic from which the keyword is chosen from. This can be calculated using the joint probability distribution of $p(w, z, s)$ (joint term by topic by sentence matrix). The value $p(w, z, s)$ gives a measure of the probabilistic co-occurrence count of a word, a topic and a sentence according to the PLSA model.

Sentence Scoring

The feature selection stage defines the events (w_g, z_g, s_n) for a sentence s_n, where $g = 1,2, \dots, L \times K$. The probability of each event gives a measure of the occurrence of a word w_g in a sentence s_n in the sense of topic z_g. If we consider the probability that all events happen for sentence s_n, then this gives us a measure of sentence s_n containing all of the selected keywords in the sense of the topics from which they were selected from. The probability of all events for the sentence s_n, $p(AllEvents_n)$, then serves as a score for that sentence. Using the *bag-of-words* assumption of the PLSA model, the score given to a sentence s_n becomes (equation 3.6 below),

$$score(s_n) = p(AllEvents_n) = \prod_{g=1}^{L*K} p(w_g, s_n, z_g) = \prod_{g=1}^{L*K} p(z_g)p(w_g|z_g)p(s_n|z_g) \quad (3.6)$$

where we have used the symmetric model of PLSA to write $p(w, s, z) = p(z)p(w|z)p(s|z)$. The two terms $p(z_g)$ and $p(w_g|z_g)$ of equation 3.6 are common to all sentences for each topic z_g and for each corresponding word w_g. This causes the score to degenerate to equation 3.7 below,

$$score(s_n) = \prod_{k=1}^{K} p(s_n|z_k) \tag{3.7}$$

Thus, JWTS assigns as a score to a sentence the product of the points of the likelihood function of z with s acting as a parameter.

Pseudocode

JWTS can be implemented by following each of the stages described above. But, because the sentence scores are given by the simpler form of equation 3.7, we use this equation for the implementation. The architectural representation of JWTS in figure 3.1 helps only for comparison purposes with the other algorithms. The pseudocode for JWTS is given in algorithm 3.1 below:

Algorithm 3.1: Sentence scoring with JWTS algorithm

Input
- PLSAParameters
- termBySentence matrix

1. **for** each sentence s_n **do**
   sentenceScores(n)= $\prod_{k=1}^{K} p(s_n|z_k)$
2. **end for**
3. predictiveLogLikelihood = logL(PLSAParmeters, termBySentence)

Output
- sentenceScore
- predictiveLogLikelihood

$PLSAParameters$ is a variable that contains all of the parameters of the PLSA model:

- $p(s)$
- $p(w|z)$
- $p(s|z)$
- $p(z|s)$ and
- $p(z)$.

$sentenceScores$ is an array that holds the scores for all sentences. $logL$ is a function that calculates the predictive log-likelihood of the document given the generated PLSA model at the topic identification stage. We keep track of the log-likelihood of the PLSA model that generated this specific sentence scores for use in the score averaging stage (see section 3.6.4 for details).

Discussion

From equation 3.7 we conclude that JWTS favors sentences that are considered important by many topics than just by few topics. This algorithm is related to algorithm PROC1 of (Bhandari et al. 2008). PROC1 uses only $p(s|z)$ of the dominant topic to create sentence scores. As pointed out by the authors, this does not help to create better generic summaries. However JWTS considers the importance score given by each topic of the document to a sentence.

JWTS is also related to algorithm PROC3 of (Bhandari et al. 2008). If we use the asymmetric model of PLSA and write $p(w,s,z) = p(s)p(z|s)p(w|z)$, the score assigned by JWTS to sentences takes the form given by equation 3.8 below

$$score(s_n) = \prod_{k=1}^{K} p(s_n)p(z_k|s_n) = [p(s_n)]^K \prod_{k=1}^{K} p(z_k|s_n) \quad (3.8)$$

PROC3 uses $p(s)$ to rank sentences while JWTS increases the influence of this parameter by raising it to the power of the $number\ of\ topics$ and weights the resulting number with the product of the points of the probability distribution of topics given a sentence.

JWTS is created to make appropriate use of terms when the terms are selected based on their ranks from $p(w|z)$, the term by topic matrix. TopicLSA simply folds the resulting terms into the latent sentence representations. As discussed in section 3.1, this amounts to comparing sentences with some low scoring words. In contrast to this, JWTS preserves the sense of the word identified by the topic it is selected from when creating the sentence scores.

3.6.1.2. CWTS Algorithm

Keyword Selection

The keyword selection strategy of CWTS is the same as JWTS. The main contents of the document are represented by selecting L high ranking terms from each topic using the distribution $p(w|z)$. The total number of selected keywords becomes $L \times K$, where K is the number of topics.

Feature Creation

Here, we assign a set of features to each sentence of the document. As in JWTS, a single sentence is assigned a total of $L \times K$ number of features. A feature corresponds to a single keyword and the topic from which it is chosen from. Instead of using $p(w, z, s)$ to check for the presence of a word in a sentence in the sense of a topic, it uses $p(w, z|s)$ (conditional term by topic by sentence matrix) to create the features for a sentence. This way of creating features favors sentences that not only contain a word in the sense of a given topic but also those sentences that give extra weight to the word-topic combination at hand with respect to all other word-topic combinations of other sentences. The conditional probability over the sentence serves as a form of sentence length normalization while the keyword is counted in the sentence. Thus,

CWTS does not favor long sentences. But since JWTS uses joint probability, it favors long sentences.

Sentence Scoring

The feature selection stage defines the events (w_g, z_g, s_n) for a sentence s_n conditioned on the appearance of s_n, where $g = 1,2,\dots, L \times K$. The probability of each event gives us a measure of the occurrence of a word w_g in a sentence s_n in the sense of topic z_g given that s_n occurs. If we consider the probability that all events happen for sentence s_n, then this gives us a normalized measure of sentence s_n containing all of the selected keywords in the sense of the topics from which they were selected from. The probability of all events for the sentence s_n, $p(AllEvents_n)$, then serves as a score for that sentence. Using the *bag-of-words* assumption of the PLSA model, the score given to a sentence s_n becomes (equation 3.9 below),

$$score(s_n) = p(AllEvents_n) = \prod_{g=1}^{L*K} p(w_g, z_g|s_n) = \prod_{g=1}^{L*K} p(w_g|z_g)p(z_g|s_n) \quad (3.9)$$

where we have used the asymmetric graphical model of PLSA to write $p(w, z|s) = p(w, z, s)/p(s) = p(s)p(z|s)p(w|z)/p(s) = p(z|s)p(w|z)$. The term $p(w_g|z_g)$ of equation 3.9 is common to all sentences for each topic z_g and for each corresponding word w_g. This causes the score to degenerate to equation 3.10 below

$$score(s_n) = \prod_{k=1}^{K} p(z_k|s_n) \quad (3.10)$$

Thus, CWTS assigns as a score to a sentence the product of the points of the topic distribution of a sentence.

Pseudocode

Like JWTS, CWTS can be implemented by following each of the stages described above. But, because the sentence scores degenerate to the simpler form given by equation 3.10, we use this equation for implementation. The pseudocode for CWTS is given in algorithm 3.2 below:

Algorithm 3.2: Sentence scoring with CWTS algorithm

Input
- PLSAParameters
- termBySentence matrix

1. **for** each sentence s_n **do**
   sentenceScores(n)= $\prod_{k=1}^{K} p(z_k|s_n)$
2. **end for**
3. predictiveLogLikelihood = logL(PLSAParmeters, termBySentence)

Output
- sentenceScore
- predictiveLogLikelihood

Discussion

If we look at equation 3.8, we see that JWTS relates to CWTS in that while CWTS uses only $p(z|s)$ for ranking, JWTS also make use of $p(s)$. CWTS gives emphasis to the conditional probability of topics with respect to a sentence while JWTS gives emphasis to the likelihood function of a topic with a sentence as a parameter. CWTS favors sentences that have topic distributions that give relatively uniform importance to all topics. Sentences dominated by few strong topics are not favored.

3.6.2. Approach 2

Approach 2 consists of the algorithms JWS (joint word sentence) and CWS (conditional word sentence). It is characterized by having a keyword selection strategy where top ranking keywords of the document that are dominant across all topics are selected. It follows two different sentence scoring strategies. JWS assigns as a score to a sentence the joint probability of the keywords and the sentence. CWS assigns as a score to a sentence the conditional probability of the keywords given the sentence. Each algorithm is discussed in detail in the sections that follow.

3.6.2.1. JWS Algorithm

Keyword Selection

To give a wide coverage for the main topics of a document, Approach 1 selects words from each topic of the document independently of one another. However, JWS selects keywords that are dominant across all topics. The basic parameter of the asymmetric PLSA model, $p(w|z)$, gives us the score given by each topic for each word. This way, a single word gets K scores, where K is the number of topics. We can now combine each of the K scores for a word so as to determine whether the word is dominant across all topics of the document. One way of combining the scores is to simply sum the K scores given to the word. But this ignores the importance of the topic that generated each score. To take this into account, we weight each score of the word by $p(z)$ before summing them. This gives us the score for a word given by equation 3.11 below

$$score(w) = \sum_{z} p(z)p(w|z) = p(w) \tag{3.11}$$

As can be seen from equation 3.11, the score we created turns out to be $p(w)$, which is actually a parameter of the asymmetric model of PLSA consisting of the parameters $p(w)$, $p(z|w)$ and $p(s|z)$ (Hofmann and Puzicha 1998). These parameters can be obtained by following the same EM steps described in section 2.3.3 but reversing the role of w and s. Thus, $p(w)$ is simply the total number of appearances of a word in a document divided by the total number of words in the document. $p(w)$ is not convenient to use as rank for words because many words are likely to share the same counts. This requires breaking quite a number of ties when ranking words. To overcome this problem, we give further importance to a topic by raising it to an integer power, α. This makes the score for a word to be different from the empirical distribution $p(w)$ and no two words to have the same score. The new score becomes (equation 3.12 below)

$$score(w) = \sum_{z} [p(z)]^{\alpha} p(w|z) \tag{3.12}$$

After assigning ranks to words based on equation 3.12, we select L words that have the highest scores.

Feature Creation

This stage assigns a set of features to each sentence of the document. For a single sentence, L number of features are calculated. The feature measures the degree to which the sentence contains the keyword. This can be calculated using the joint probability distribution of $p(w, s)$ (joint term by sentence matrix). The value $p(w, s)$ gives a measure of the probabilistic co-occurrence count of a word and a sentence.

Sentence Scoring

The feature selection stage defines the events (w_l, s_n) for a sentence s_n, where $l = 1,2, \dots, L$. The probability of each event gives us a measure of the occurrence of a word w_l in a sentence s_n. If we consider the probability that all events happen for sentence s_n, then this gives us a measure of sentence s_n containing all of the selected keywords. The probability of all events for the sentence s_n, $p(AllEvents_n)$, then serves as a score for that sentence. Using the *bag-of-words* assumption of the PLSA model, the score given to a sentence s_n becomes (equation 3.13 below),

$$score(s_n) = p(AllEvents_n) = \prod_{l=1}^{L} p(w_l, s_n) = \prod_{l=1}^{L} [p(s_n)p(w_l|s_n)]$$
$$= \prod_{l=1}^{L} [p(s_n) \sum_{z} p(w_l|z)p(z|s_n)]$$
$$= [p(s_n)]^L \prod_{l=1}^{L} \sum_{z} p(w_l|z)p(z|s_n) \quad (3.13)$$

Using the symmetric graphical model of PLSA we can write

$$p(w, s) = \sum_{z} p(z)p(w|z)p(s|z) = \sum_{z} p(w, z)p(s|z) \quad (3.14)$$

This expression can also be used in equation 3.13. Equation 3.14 shows us that the score of equation 3.13 cannot be simplified further because the term which is common to all sentences, $p(w_l, z)$, is coupled with other terms using addition.

Pseudocode

Unlike the implementations of Approach 1, JWS needs to be implemented by following each of the stages described above since its score does not degenerate to a simpler form. The pseudocode for JWS is given in algorithm 3.3 below:

Algorithm 3.3: Sentence scoring with JWS algorithm

Input
- PLSAParameters
- termBySentence matrix
- L (Number of Keywords)
- α (Used for term ranking)

1. **for** each unique word in the document w_m **do**
   wordScores(m)= $\sum_z [p(z)]^{\alpha} p(w_m|z)$
2. **end for**
3. Choose L high scoring words from wordScores and use them as keywords
4. **for** each sentence s_n **do**
   sentenceScores(n)= $\sum_{l=1}^{L} \ln (p(s_n) \sum_z p(w_l|z) p(z|s_n))$
5. **end for**
6. predictiveLogLikelihood = logL(PLSAParmeters, termBySentence)

Output
- sentenceScores
- predictiveLogLikelihood

As can be seen from step 4, we calculated the natural logarithm of equation 3.13 for implementation purposes. Hence, the product of $p(w_l, s_n)$ becomes a summation. The reason we use logarithm is that because the number of keywords would be quite large and the value $p(w_l, s_n)$ would be very small, the score for the sentence would evaluate to zero since it would be too small for the computer to hold.

Discussion

JWS is related to PROC3 of (Bhandari et al. 2008). PROC3 uses $p(s)$ as sentence rank while JWS uses $[p(s)]^L$ and weights it with sentence-specific word distributions, $p(w|s)$, of the selected keywords. JWS is similar to JWTS since both use joint probabilities for scoring. We also note that the use of $p(w, s)$ for sentence ranking makes use of all senses/topics of a word as can be seen from equation 3.13. This is not a problem here since the keyword extraction procedure also considers all topics of a word during sentence ranking.

3.6.2.2. CWS Algorithm

Keyword Selection

The keyword selection strategy of CWS is the same as JWS. The main contents of the document are represented by selecting L high ranking terms, where equation 3.12 is used for giving scores to words of the document.

Feature Creation

This stage assigns a set of features to each sentence of the document. As in JWS, a single sentence is assigned a total of L number of features. The feature measures the degree to which the sentence contains the keyword. Instead of using $p(w, s)$ to check for the presence of a word in a sentence, it uses $p(w|s)$ (conditional term by sentence matrix) to create the features for a sentence. This means that CWS measures the relative occurrence/importance of a word with respect to other words in the sentence.

Sentence Scoring

The feature selection stage defines the events (w_l, s_n) for a sentence s_n conditioned on the appearance of s_n, where $l = 1,2, \dots, L$. The probability of each event gives us a measure of the occurrence of a word w_l given that sentence s_n occurs. If we consider the probability that all events happen for sentence s_n, then this gives us a normalized measure of sentence s_n containing all of the selected keywords. The probability of all events for the sentence s_n, $p(AllEvents_n)$, then serves as a score for that sentence. Using the *bag-of-words* assumption of the PLSA model, the score given to a sentence s_n becomes (equation 3.15 below),

$$score(s_n) = p(AllEvents_n) = \prod_{l=1}^{L} p(w_l|\, s_n) = \prod_{l=1}^{L} \sum_{z} p(w_l|z)p(z|s_n) \qquad (3.15)$$

The score of equation 3.15 cannot be simplified further because of the coupling between different values of $p(w|z)$ using addition.

Pseudocode

Like JWS, CWS can be implemented by following each of the stages described above. The pseudocode for CWS is given in algorithm 3.4 below:

Algorithm 3.4: Sentence scoring with CWS algorithm

Input
- PLSAParameters
- termBySentence matrix
- L (Number of Keywords)
- α (Used for term ranking)

1. **for** each unique word in the document w_m **do**
   wordScores(m)= $\sum_z [p(z)]^{\alpha} p(w_m|z)$
2. **end for**
3. Choose L high scoring words from wordScores and use them as keywords
4. **for** each sentence s_n **do**
   sentenceScores(n)= $\sum_{l=1}^{L} \ln \left(\sum_z p(w_l|z)p(z|s_n)\right)$
5. **end for**
6. predictiveLogLikelihood = logL(PLSAParmeters, termBySentence)

Output
- sentenceScores
- predictiveLogLikelihood

Discussion

Looking at the third row of equation 3.13, we see that JWS relates to CWS in that while CWS uses only $p(w|s)$ for ranking, JWS also makes use of the sentence length $p(s)$. CWS is similar to CWTS in that that both use conditional probabilities given a sentence. That is, they normalize sentence length when forming sentence scores.

3.6.3. Approach 3

Approach 3 consists of the algorithms KITS (keywords in topic simplex) and KIVSM (keywords in vector space model). Its keyword selection strategy is like that of approach 2 where top ranking keywords of the document that are dominant across all topics are selected. Unlike the keyword selection strategy used by Approach 1, the keyword selection procedure used here lends itself to another method of sentence ranking algorithm, namely, representing the selected keywords in the same manner as the sentences in a space (topic simplex in the case of KITS and VSM in the case of KIVSM) and selecting sentences that are most similar to the keywords. Each algorithm is discussed in detail in the sections that follow.

3.6.3.1. KITS Algorithm

Keyword Selection

The keyword selection strategy of KITS is the same as the one used in Approach 2. Equation 3.12 is used to select *L* high ranking terms of the document so as to represent the main points of the document.

Feature Creation

This stage assigns features to the sentences of the document and the keywords. This feature is given by the representation of the sentences and documents in the topic simplex. A sentence is represented by $p(w|s)$, which is a point in the topic simplex. In the PLSA model, this is given by equation 2.10 of section 2.3.3. To represent the keywords in the same manner as the sentences in the topic simplex, we concatenate the keywords as one sentence and treat them as a query, q. Then this query is folded into the topic simplex (see section 2.3.3) to give a representation of the query, $p(w|q)$, by following the procedure put forward by (Hofmann 1999b).

Sentence Scoring

The folding process we used in the feature selection step gives us a representation of the keywords in the topic simplex, $p(w|q)$, which is obtained from the topic mixture of the keywords estimated during folding, $p(z|q)$. This topic mixture is characteristic of the identities of each keyword. In a similar fashion, the representation of a sentence in the topic simplex,

$p(w|s)$, is characteristic of the words it contains as estimated from the PLSA model. Thus, comparing the representations of sentences, $p(w|s)$, and keywords, $p(w|q)$, by similarity measures such as the KL (Kullback Leibler) divergence, JS (Jensen-Shannon) divergence and cosine (section 2.3.4) amounts to another way of measuring term overlap between sentences and the keyword. The score for a sentence would then be its similarity with q as measured by any one of the three similarity measures mentioned above.

Pseudocode

The pseudocode for KITS is given in algorithm 3.5 below.

Algorithm 3.5: Sentence scoring with KITS algorithm

Input

- PLSAParameters
- termBySentence matrix ($M \times N$)
- L (Number of Keywords)
- α (Used for term ranking)
- similarityMeasure (can have the values of KL, JS, or cosine)
- iterations (the number of iterations to be performed during folding)
- βf (tempering parameter used for folding)

1. **for** each unique word in the document w_m **do**
   wordScores(m)= $\sum_z [p(z)]^{\alpha} p(w_m|z)$
2. **end for**
3. Choose L high scoring words from wordScores and use them as keywords
4. Form the keywords query, Query. Query is an $M \times 1$ matrix. It has a one entry at the index of the keywords (index is according to the input termBySentence matrix) and a zero entry elsewhere.
5. $p(z|q)$ = fold(Query, $p(w|z)$, iterations, βf)
6. $p(w|q)$ = $\sum_z p(w|z)p(z|q)$
7. **for** each sentence s_n **do**
   $p(w|s_n)$ = $\sum_z p(w|z)p(z|s_n)$
8. **end for**
9. **for** each sentence s_n **do**
   sentenceScores(n)= similarity between $p(w|q)$ and $p(w|s_n)$ measured using similarityMeasure
10. **end for**
11. predictiveLogLikelihood = logL(PLSAParmeters, termBySentence)

Output

- sentenceScores
- predictiveLogLikelihood

In algorithm 3.5, $fold$ is a function that performs the folding operation.

Discussion

KITS is similar to TopicLSA – Genre because it uses the low dimensional representation provided by PLSA. However, it is different by its method of keyword extraction. Unlike TopicLSA - Genre, the ranking procedure of KITS is in line with its keyword extraction procedure. Because a keyword is selected by considering all of its topics, folding it into the topic simplex will not be a concern for KITS. Algorithms of Approach 1 cannot use this folding approach without modifications because the folding process adds senses of words for which they were not selected for.

3.6.3.2. KIVSM Algorithm

Keyword Selection

KIVSA selects keywords of the document in the same manner as KITS. It uses equation 3.12 to select *L* high ranking terms of the document so as to represent the main concepts of the document.

Feature Creation

KIVSM uses the vector space model (VSM) to create features for the sentences of the document and the keywords that result from the keyword selection phase. It constructs a VSM from the original term by sentence matrix of the document taking the terms as the axes of the space. A sentence is then assigned to a point in the resulting space according to its coordinates which is given in the original term by sentence matrix of the document. This point will serve as a feature for the sentence. To assign a feature to the keywords, they are first treated as one sentence. Then, this keywords sentence (query) is plotted just like a sentence in the VSM space, associating it with a single point. This point will serve as a feature that characterizes the keywords.

Sentence Scoring

Just like the sentence scoring phases for the other algorithms, KIVSM tries to measure the term overlap between the keywords and each sentence. Since the sentences and the keyword are found in a VSM space, we calculate the similarity between a sentence and the keywords query using either the cosine or dot product similarity measures. This similarity between a sentence and a

query will serve as the sentence score. When KIVSM is used with the cosine similarity, we can label it as KIVSM-cos, and when used with the dot product, we can label it as KIVSM-dot.

Pseudocode

The pseudocode for KIVSM is given in algorithm 3.6 below.

Algorithm 3.6: Sentence scoring with KIVSM algorithm

Input
- PLSAParameters
- termBySentence matrix ($M \times N$)
- L (Number of Keywords)
- α (Used for term ranking)
- similarityMeasure (can have the values of cosine or the dot product)

1. **for** each unique word in the document w_m **do**
   wordScores(m)= $\sum_z [p(z)]^{\alpha} p(w_m|z)$
2. **end for**
3. Choose L high scoring words from wordScores and use them as keywords
4. Form the keywords query, Query. Query is an $M \times 1$ matrix. It has a one entry at the index of the keywords (index is according to the input termBySentence matrix) and a zero entry elsewhere.
5. **for** each sentence s_n **do**
   sentenceScores(n)= similarity between a column of the termBySentence matrix corresponding to sentence s_n and Query measured using similarityMeasure
6. **end for**
7. predictiveLogLikelihood = logL(PLSAParmeters, termBySentence)

Output
- sentenceScores
- predictiveLogLikelihood

Discussion

This algorithm is similar to KITS except for the fact that it uses VSM to represent sentences and keywords. We have created this algorithm to assess the advantages of using the PLSA model by the JWS, CWS and KITS algorithms to estimate the presence of keywords of the document in the various sentences.

JWS, CWS and KITS make use of the resulting PLSA model to reliably estimate the degree to which the keywords are contained in each sentence. JWS and CWS make use of $p(w,s)$ and $p(w|s)$ respectively to score sentences. As shown in equations 3.13 and 3.15, both $p(w,s)$ and $p(w|s)$ are calculated from parameters of the PLSA model rather than the empirical distributions

of $\hat{p}(w,s) \equiv n(w,s)/\sum_{w,s} n(w,s)$ and $\hat{p}(w|s) \equiv n(w,s)/n(s)$. In a similar fashion, KITS makes use of $p(w|q)$ as estimated from the PLSA model. In contrast to this, KIVSM makes use of the sparse original term by sentence matrix. But this algorithm makes use of the same keyword extraction method as JWS, CWS and KITS. For this reason, KIVSM can be called a half-concept-based approach.

3.6.4. Score Averaging

At the heart of creation of the PLSA model is the Expectation Maximization (EM) algorithm (section 2.3.3). The EM algorithm tries to find the local maximum of the likelihood function, $\mathcal{L}$, of the document to be summarized. EM proceeds by starting from random initial values for parameters of the PLSA model ($p(s|z)$ and $p(w|z)$) and iteratively recompute these parameters to converge at a local maximum. Because there are many local maxima for $\mathcal{L}$, different runs of the PLSA model results in different model parameters. To overcome this issue (Brants et al. 2002), in their document segmentation work, propose to generate five different scores using five different runs of the EM algorithm and average the results to get the final score. Their work shows that results are improved when compared to a system that uses only one run of the EM algorithm.

The algorithms that have been discussed in sections 3.6.1, 3.6.2 and 3.6.3 can be implemented without using this step. But we expect that they would suffer from relatively poor summarization performance when compared to the implementation that uses the score averaging component. Following (Brants et al. 2002), we also use model averaging to help improve the performance of the proposed summarization system. Thus, what the score averaging component does is, it accepts scores from different runs of a sentence ranking algorithm, averages the scores for the different runs and finally assigns this value as the final score to the sentences. The model averaging scheme we follow is detailed in section 4.3.1.

3.6.5. Sentence Selection

The last task performed in the sentence ranking sub-system is sentence selection. It selects sentences that have the highest score according to the scores that result from the score averaging component. In addition to this, we took advantage of the fact that we experiment with news articles. It has been observed that the first few sentences of a news article bear important information (Nenkova 2005). To use this fact in our summarization algorithms, we follow the work of (Ledeneva 2008). Ledeneva modifies her ranking algorithms in such a way that k best+first sentences are always selected, where k best means the k top ranked sentences according to her proposed ranking algorithms. This means that when she wants to select S sentences for the summary, she selects $S - k$best number of first sentences of the document and then select $k$best sentences from her proposed ranking algorithms. The best summarizer is found when $k = 1$, i.e., only one sentence is chosen using her proposed algorithm while the rest of the sentences come from the first sentences of the document.

In contrast to this, we plan to use only the first sentence of the document as the sentence that needs to be always included in the summary. The rest of the sentences would come from our ranking algorithms. This may not give the maximum performance for our system, but instead we would like to investigate the strength of the proposed ranking algorithms by giving them minimum support.

To sum up, the steps followed during sentence selection are:

1. Accept sentence scores from the score averaging component.
2. Sort the sentence scores.
3. Pick the top ranking sentences. The number of top ranking sentences selected is given by $extraction\ rate \times total\ number\ of\ sentences\ in\ the\ document.$
4. If the first sentence has not been selected at step 3, it is included at the expense of the lowest ranking sentence that has been selected as a summary sentence in step 3.

3.7. Summary

In this chapter, we proposed six new keyword-based text summarization algorithms using PLSA. The six algorithms are divided into three approaches. Approach 1: consisting of algorithms JWTS and CWTS. Approach 2: consisting of JWS and CWS. Approach 3: consisting of KITS and KIVSM.

Unlike most of the previous works (except TopicLSA by Melese 2009), the algorithms make use of the term by concept matrix that result from topic modeling. All of the proposed algorithms have two major steps: keyword selection and sentence ranking. Keywords of the document are selected by making use of the term by concept matrix. The keywords represent the main points of the text to be included in the summary. The sentence ranking step estimates the presence of the selected keywords in each sentence. Sentences that best contain the selected keywords are selected to form the document summary. Since we experiment with news articles, the sentence ranking algorithm is designed to always select the first sentence of the document in order to take advantage of the fact that important information exists near the beginning of news articles most of the time.

Our work is different from TopicLSA in two major ways:

1. While TopicLSA uses LSA to represent topics of the document, we use PLSA. PLSA suits discrete data such as text documents and is shown to outperform LSA based systems in many experiments. This way, our algorithms take advantage of topics that have been identified with PLSA which has been specifically designed for discrete data.
2. We identified TopicLSA as a keyword approach and showed that its keyword ranking and sentence ranking algorithms are not in tune. In contrast to this, the algorithms we proposed have sentence ranking algorithms that make appropriate use of the selected keywords.

In order to fulfill point 2 cited above, we proposed to use two different keyword selection strategies. The first strategy has been used by TopicLSA; it selects top ranking terms from each topic/concept as identified from the PLSA model. Approach 1 uses this strategy. Folding the

resulting keywords in the topic simplex, like in TopicLSA, and using it for ranking will not yield a sound sentence ranking algorithm for the keyword selection strategy of Approach 1. Instead, we propose to use a ranking algorithm that selects a sentence that has a high value for the joint/conditional probability of the keywords, topics of the keywords from which they were selected for and the sentence. When the joint probability is used, we get the ranking algorithm JWTS. When the conditional probability is used, we get the ranking algorithm CWTS. In information retrieval applications, the conditional probability measure is usually used so as to give equal chance for a long and short sentence to be selected. But the requirement could be different for the task of text summarization and hence we want to see the application of the joint probability measure to the task at hand.

To use the folding approach like TopicLSA, we proposed the second keyword selection strategy that selects keywords by considering their score with respect to each topic. This way, we can use the folding approach. The resulting algorithm is KITS of Approach 3. Approaches 2 and 3 make use of this keyword selection strategy. We also considered ranking algorithms that are similar to algorithms of Approach 1 when ranking. This results in Approach 2. Approach 2 selects a sentence that has a high score for the joint/conditional probability of the keywords and a sentence. When joint probability is used, we get JWS. When conditional probability is used we get CWS. Finally, to investigate whether detecting keywords from the reliable PLSA representation is really advantageous when compared to the original sparse term by sentence matrix, we proposed the KIVSM algorithm of Approach 3. This algorithm is similar to KITS. While KITS uses the topic simplex to represent keywords as query, KIVSM uses the vector space model of the original sparse term by sentence matrix to represent keywords as query.

Chapter 4. Evaluation

This chapter presents the evaluation we carried out on the summarization methods we proposed in chapter 3. Section 4.1 presents implementation details such as libraries of modules used to implement the proposed methods and the way the summarizers are implemented for the evaluation purpose. The data corpus used for evaluation, the evaluation procedure we followed and discussion of systems implemented by other authors used for comparison purposes with the proposed algorithms are covered in section 4.2. In Section 4.3, we present in detail the experimental setup used, results of the experiments and discussion of the results. Section 4.4 presents an overall discussion of our experimental findings.

4.1. Implementation Details

We implemented the various algorithms we proposed for Amharic text summarization in order to evaluate their performance.

As pointed out in section 3.4, we used the pre-processing module as adopted/implemented by (Melese 2009) from the work of (Tessema 2007). We also used Melese's term by sentence construction module in our work. The pre-processing module makes use of, Lucene[6], which is an open-source information retrieval library implemented in Java. We have used JAMA[7] for making computations associated with matrices since it makes the calculations much easier to implement. We have used MathWork and NIST's reference implementation of JAMA.

Since we carry out a large number of experiments, we have made the evaluation process fully automated. Because of this, the architecture of the summarizer contains a component, labeled "evaluation on precision/recall", which is used for evaluation purposes. Thus, the summarizer calculates precision/recall for each summarized document according to fixed parameters as per the requirements of the experiment being run, then calculates the average precision/recall for all the documents and finally writes the result in a file. Typically, in a single run for a given set of parameters, 135 or 160 files are produced. So, java code is written to read the resulting

[6] http://lucene.apache.org
[7] http://math.nist.gov/javanumerics/jama

precision/recall results from the files, which are later transferred to Microsoft Excel for further analysis.

Each of the six algorithms that are evaluated has its own class. In order to make the implementation modular and make code reuse between each classes easier, we have used three methods for each class. Each class contains the methods:

- *rank*
- *score*
- *exp*

rank produces scores for each sentence of the document using a single PLSA model for a fixed set of parameters. The method *score* is used to create the final score for each sentence by summing scores that result from calling *rank* repeatedly for a number of random restarts of the PLSA model. *exp* accepts list of summary sentence numbers from *score*, calculates precision/recall for each document, averages the precision/recall from all documents and finally writes the resulting average value to a file. As a sample, we have put the java class for JWTS in Appendix E.

We experiment for the three extraction rates of 30, 25 and 20%. The required extraction rate is passed only to score and exp. score uses the extraction rate to limit the number of sentences it returns to exp. exp uses the extraction rate value as follows: if the extraction rate is 30%, it generates precision/recall results for 25 and 20% as well, otherwise only the indicated extraction rate has its precision/recall result calculated. During all of our experimentation, we have passed the extraction rate of 30% to *exp* (and hence to *score*) in order to limit the running time for the experiments. When a document is summarized, *score* always selects the top ranking sentences according to the extraction rate. For example, if a document has a total of 30 sentences and the extraction rate is 30%, then $30 \times 0.3 = 9$ sentences are extracted. Once these 9 sentences reach *exp*, it reduces them to a length required for producing 25% (8 sentences) and 20% (6 sentences) rate summaries. This way, we have reduced the total running time of our experiments by two-thirds.

4.2. Experimental Settings

This section describes the preparations made to carry out experiments that test the performance of the proposed summarization systems.

4.2.1. Data Corpus

60 Amharic news articles are used for the experiment. 47 of the documents are acquired from Melese's work (Melese 2009), while we have added 13 new documents, taken from the Ethiopian Reporter website[8]. Table 4.1 below provides the statistics of the corpus.

Table 4.1: Corpus statistics.

Corpus Attribute	Value
Number of documents	60
Minimum number of sentences per document	10
Maximum number of sentences per document	45
Average number of sentences per document	27
Minimum number of words per document	382
Maximum number of words per document	933
Average number of words per document	585

Three independent human evaluators were used to generate ideal summaries for the 60 documents against which the summaries produced by our systems are compared. Thus, a single document has three summaries, one per each evaluator. Each evaluator was asked to select sentences that s/he considers the best that summarizes the content of each document. The number of sentences selected at a given extraction rate is obtained by multiplying the total number of sentences of the document without the title by the extraction rate and rounding off the result to the nearest integer. The evaluators were asked to generate summaries for the three extraction rates of 30%, 25% and 20%. Once they select the sentences for the 30% extraction rate, they were asked to eliminate the required number of sentences from those selected for 30% to reach at

[8] http://www.ethiopianreporter.com

25% extraction rate summaries. In a similar fashion, to get summaries for 20% extraction rate, the required number of sentences is removed from those selected for the 25% extraction rate.

The work of (Brown and Day 1983) identifies six basic rules of generating abstractive summarization. Because the ideal summaries we require are extractive, only two of these rules are relevant to our work. The rules we use are

1. Deletion of material that is trivial
2. Deletion of material that is redundant even if it is important.

These two rules were given to the human summarizers in addition to other instructions to prepare the ideal summaries. See Appendix A for the guideline and instructions we used for the preparation of ideal summaries.

4.2.2. Evaluation Procedure

In order to evaluate the performance of the summarization systems, we have used the precision/recall/F-measure metric (see section 2.2.1). This metric is used because our summarization systems produce extractive summarization and the ideal summaries are also composed of whole sentences extracted from the summarized document.

Because the number of sentences chosen by both humans and the system are always the same, equations 2.1, 2.2 and 2.3 give the same result. This means that precision, recall and F-measure have all the same values. To reflect this, we have indicated the performance results as "precision/recall" in all cases.

A given summarization algorithm is evaluated for precision/recall according to the following steps:

1. Calculate the precision/recall obtained by summaries of the algorithm for all documents with respect to each ideal summary.

2. Sum up all of the resulting precision/recall values and divide it by $60 \times 3 = 180$. This gives the average precision/recall attained by the algorithm with respect to the 60 documents and the three human evaluators.

4.2.3. Compared Systems

There are a number of single-document summarization systems developed by different scholars for different languages. We have selected systems which are implemented using topic modeling approaches (LSA and PLSA) that are reported to have performed the best. TopicLSA is selected to represent systems that are implemented using LSA, while PROC3 is used to represent systems that are implemented using PLSA. We have not found a single-document text summarization system that has been implemented using LDA. We have included two additional summarizers (random summarizer and first *n* sentences summarizer) that are used as baselines. See table 4.2 for the performance of these summarizers. The performance of each of the above summarizers has been tested using our Amharic corpus.

Random Summarizer

This summarizer selects summary sentences randomly. Following the work of (Steinberger and Ježek 2004; Ledeneva 2008), we report the average of 10 runs of the random summarizer.

First *n* Sentences

This summarizer gives the highest importance to the first sentence, then the second sentence and so on, ranking the last sentence of the document the lowest. Depending on the extraction rate and the number of sentences of the document, this summarizer selects the first *n* sentences of the document ($n = extraction\ rate \times number\ of\ sentences$). We chose this summarizer because it is considered a very strong baseline in the domain of news articles (Nenkova 2005). This is because of the journalistic tendency of putting the most important points of an article in the initial paragraphs.

TopicLSA

This is a system by (Melese 2009) discussed in sections 2.4.5 and 3.1. Experiments by (Melese 2009) have shown that TopicLSA outperforms other LSA based single-document summarization systems. To get the precision/recall score for TopicLSA we have run the algorithm for the 16 term weighting functions and six number of terms (3, 5, 6, 8, 10, and 15) as specified in (Melese 2009). This means that we have $16 \times 6 = 96$ system variations for the algorithm. The scores shown in table 4.2 are the maximum for each extraction rate out of the 96 system variations.

PROC 3/PROC 3+

PROC 3 is a system by (Bhandari et al. 2008) discussed in section 2.4.5. It is the best reported system using PLSA for single-document summarization. We have also created a new algorithm by modifying the original algorithm in such a way that it always includes the first sentence of the document, giving rise to the PROC3+ algorithm. In this manner, we can compare our work with that of (Bhandari et al. 2008) more interestingly since some of our algorithms relate to PROC3. As discussed in section 2.4.5, this system uses $p(s)$ to rank each sentence. However, $p(s)$ is simply the normalized word count of a sentence (without stop words), as discussed in section 2.3.3. Thus, in other terms, PROC3 can be called *Longest Sentences* summarizer.

PROC3 is implemented in (Bhandari et al. 2008) using $p(s) = \sum_z p(s|z)p(z)$, where the parameters $p(s|z)$ and $p(z)$ are from the symmetric graphical model of PLSA. However, this implementation is prone to the randomized initial conditions of the PLSA model as far as getting the true word count of a sentence, $p(s)$, is concerned. This can be verified from their work where different runs of PROC3 for various numbers of topics give different results in ROUGE scores. The highest score is achieved when the number of topics is 2. In contrast, we use the asymmetrical model of PLSA and are able to get the accurate value of $p(s)$ so that there is no need to carry out experiments on PROC3 for the various parameters of the PLSA model such as the number of topics. In fact, under the asymmetric model of PLSA, the value of $p(s)$ is independent of all parameters of the PLSA model.

Because of the way our sorting algorithm is implemented, if we have two sentences that have the same length (word count), then the sentence that appears first in document gets selected. This is a

desirable behavior since news reports tend to put important information in the beginnings of the document rather than at the end.

Table 4.2 below gives the precision/recall value for each of the systems discussed above.

Table 4.2: Precision/Recall values for compared systems.

Compared System	**Precision/Recall**		
	Extraction Rate (%)		
	30	**25**	**20**
Random Summarizer	0.30973	0.26208	0.20345
First *n* Sentences	0.45770	0.45675	**0.46689**
TopicLSA	0.43946	0.41611	0.37202
PROC3	0.44956	0.39045	0.30935
PROC3+	**0.51383**	**0.46147**	0.42990

As table 4.2 shows, the best systems are PROC3+, for 30% and 25%, and First *n* Sentences, for 20%. We can see from the table that addition of the first sentence to PROC3 gives a big difference in the scores. PROC3 performs better than TopicLSA at 30% but is outperformed at 25% and 20% extraction rates. The Random summarizer is the least performing system and is outperformed by all the other systems by a big margin. It can be observed from the table that for the kind of texts we experimented with (news articles), the First *n* Sentences heuristic always performs better than the Longest Sentences (PROC3) heuristic. The difference becomes more significant at lower extraction rates.

4.3. Experiments

This section describes the way we designed our experiments to evaluate the performance of the proposed summarization algorithms and presents the results. Our summarization algorithms have a number of parameters. To account for this, we have designed our experiments so that they would be manageable in size. Section 4.3.1 discusses the parameters and the values they have. We have carried out six experiments in total, one for each algorithm. The experiments are presented from section 4.3.2 up to 4.3.9.

4.3.1. Setting System Parameters for the Experiments

Scoring Method

As discussed in section 3.6.4, we use score averaging to help improve our PLSA-based summarizer. In order to take this into account, we define a parameter called *scoring method*. We define scoring method to be which of the models of PLSA that result from different runs of the EM algorithm are used to create the average score for a sentence. It is to be noted that each PLSA model is obtained from random initial conditions.

To counter act the randomized initial conditions of the EM (Expectation Maximization) algorithm during creation of a PLSA model and hence achieve better precision/recall result, we have explored 13 different scoring methods. In his information retrieval experiments, Hofmann (Hofmann 2001) observed that the global maximum and local maxima produce models that have low performance when compared to other models that are obtained by running the PLSA algorithm. We use this finding in the following manner. Out of, say, 20 restarted PLSA models which of the models should we use? The models that have the best likelihood values, $\mathcal{L}$ (which are possible local maxima), or the models that have "poor" $\mathcal{L}$? Thus we define the Position parameter as being *top*, *bottom* or *all*. *Top* means that we use the top models out of all of the restarted models, say 5 out of a total of 20. *Bottom* refers to models with lower values for $\mathcal{L}$ including the lowest model in terms of $\mathcal{L}$ and *all* means that we use all the generated models. In the works of (Brants 2002; Hennig 2009), 5 different models are generated to get an average scores. In the hope of getting better results, we explore the use of 1, 5, 10, 15 and 20 different PLSA models. We have limited the maximum value to 20 because of compromise made with respect to computational time. When these numbers of models are combined with the three position parameters, we get 13 different scoring methods, shown in table 4.3.

Table 4.3: Scoring Methods

Position	Number of Models	Scoring Method
All	1	1 random start
	5	5 restarts[9]
	10	10 restarts
	15	15 restarts
	20	20 restarts
Top	1	Top 1 of 20 restarts
	5	Top 5 of 20 restarts
	10	Top 10 of 20 restarts
	15	Top 15 of 20 restarts
	20	Top 20 of 20 restarts (same as 20 restarts)
Bottom	1	Bottom 1 of 20 restarts
	5	Bottom 5 of 20 restarts
	10	Bottom 10 of 20 restarts
	15	Bottom 15 of 20 restarts
	20	Bottom 20 of 20 restarts (same as 20 restarts)

We have avoided scoring methods such as top 5 of 10 restarts. This is because of our assumption that more number of models help better in the exploration of good models than small number of models. The Position parameter of *all* is meant to represent a mix of models with high, medium and low values of $\mathcal{L}$.

Tempering Parameter, β, for Generating the PLSA Model

This value is used during the construction of the PLSA model via the EM algorithm (see section 2.3.3). According to (Hofmann 2001), β falls in the range of 0.6 to 1(inclusive). We have also run experiments on three representative documents[10] (small, medium and large sized documents) from our data set to see the effect of β on the resulting PLSA models. We have found that for values of β below 0.6, the resulting models of $p(w|z)$ for different topics are almost equal. Because of this we have limited the value of β between 0.6 and 1. According to the tempered EM (TEM) algorithm of (Hofmann 2001), β has the maximum value of 1, which is considered as

[9] Note that for all scoring methods, "restarts" refers to independently generated models, each generated from random initial conditions.
[10] These three documents are used throughout our experiments when the need arises.

the unmodified EM algorithm. The actual values of β used in our experiments are 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, and 1.

Tempering Parameter, βf, for Folding

During the process of folding into a trained PLSA model we may use either EM (Brants et al. 2002; Monay 2007) or TEM (Hofmann 1999b). We have carried out some experiments on three representative data sets. In our work, the main reason we used TEM in the training of a PLSA model is to reduce the effect of local maxima during convergence. The folding operation proceeds by minimization of the KL divergence between the query and points of the topic simplex (Hofmann 1999b). According to our experiments, we have observed only one maximum during the folding operation. For this reason, the value of βf has been fixed to 1.

Convergence criterion (%) for Generating the PLSA Model

The iterative EM algorithm performs a maximization process while finding parameters of the PLSA model. The question here is when do we judge to have attained enough maximum value of $\mathcal{L}$.

We have referred to the literature and found only one work that discusses the value for the convergence criterion (Brants et al. 2002). They report that an iteration of 20 of the EM algorithm usually suffices (they use a value of 1 for β througout their experiments). We conducted our own experiments to take a closer look at the behavior of the EM algorithm while iterating. For this, we used three representative documents and studied the values attained by successive values of $\mathcal{L}$ under different values of other parameters of the PLSA model such as the number of topics, β and different randomized initial conditions.

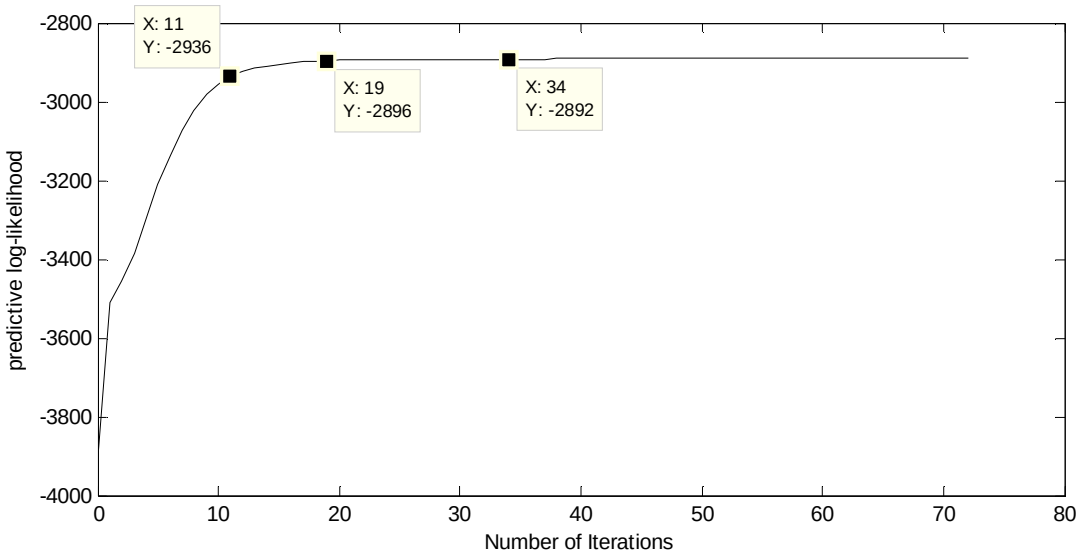

Figure 4.1: Predictive log-likelihood for a sample document (5 topics, $\beta = 0.85$).

Figure 4.1 shows the progression of $\mathcal{L}$ for a sample document when the number of topics is five and β is 0.85. From the experiments we conducted, we identified three points of interest. We call them *characteristic points* of $\mathcal{L}$. The first one is when the iteration is at 11. This point, as can be seen from the graph, is the "turning point" where $\mathcal{L}$ is just about to start to stabilize. The second point, when the iteration is 19, corresponds to the point used by (Brants et al. 2002). This point can be identified as when $\mathcal{L}$ has almost stabilized ("stabilization point"). The third point, $x = 34$, can be associated with a point ("overtraining point") that is close to a local maximum. As we go closer and closer to a local maximum like the third point does, we have what we call overtraining (Brants et al. 2002), where $\mathcal{L}$ changes by a very small margin at each iteration. Points of overtraining have shown poor performance in information retrieval experiments (Hofmann 2001).

As discussed in section 3.5, the convergence criterion is measured in percentage. The main reason for the adoption of % value is that for different documents and different parameters of the PLSA model, the number of iterations are not reliable indicators of when the characteristic points appear. For example, for low values of β such as 0.65, the stabilization point of $\mathcal{L}$ may appear when the number of iterations has reached in the 40s.

Based on our experiments with three sample documents from our data sets, we have assigned the convergence criterion for the various points:

- 0.9% for the turning point
- 0.03% for the stabilization point
- 0.007% for the overtraining point.

Depending on the document and the model parameters, the convergence criterion of 0.9% may result roughly in 5 to 15 iterations. So, increasing the convergence criterion beyond 0.9% might cause undertraining. As discussed above, the value of 0.03% is selected based on the work of (Brants et al. 2002). We have chosen the value of 0.007% to fairly represent points of overtraining, without going very near to the local maximum.

Convergence criterion (Number of Iterations) for Folding

In the information retrieval (Brants 2002) and document segmentation (Brants et al. 2002) papers, the authors write that only a very small number of iterations are sufficient for the folding operation. We also performed our own experiments based on three documents. Figure 4.2 shows the plot of the log-likelihood of a query for five number of topics, $\beta = 0.85$ and number of keywords = 20%. The plot is quite smoother than that of figure 4.1, perhaps attributable to the existence of only one local maximum. We tried to determine characteristic points for the log-likelihood by measuring the convergence criterion in % value. However, we could not get consistent % value for our candidate characteristic points. Instead, we opted for the number of iterations for measurement of the convergence criterion, which surprisingly gives quite accurate identification of properties of the log-likelihood.

As the figure shows, at about five iterations, we achieve convergence value, which is quite a small number of iterations. But, from our experiments, we saw that, to achieve a very precise point of the local maximum it usually requires a very large number of iterations, such as 100, depending on the type of document and the parameters of the PLSA model. Because we are not sure of the effect of overtraining on the performance of our summarization algorithm, KITS, we planned to experiment for a large number of iterations: 5, 10, 15, 20, 30, 40, 60, 100, and 200.

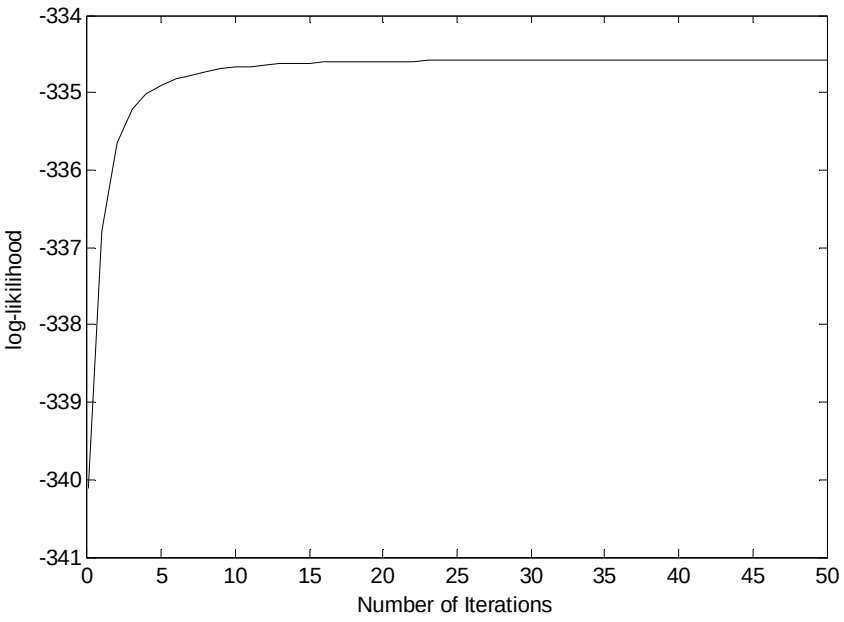

Figure 4.2: Log-likelihood for a sample document (5 topics, $\beta = 0.85$, Number of keywords=20%).

Number of Topics

The use of increasingly larger number of topics has shown a continuous improvement in performance in information retrieval experiments (Hofmann 2001). Hofmann indicates that the number of topics can even be higher than the rank of the term by sentence matrix (which is not the case of LSA). However, for the case of single-document text summarization, where the term by sentence matrix is quite sparse and the topics discussed are usually small, we believe that the number of topics 2, 3, 4, 5 and 6 give us a reasonable exploration of the effect of this parameter. The work of (Bhandari et al. 2008) also uses these number of topics for their experiments.

Term Ranking Parameter, α

For this parameter we have used the values of 2, 3, 4, and 5. Higher values are still possible but this increases the number of our experiments by a big margin. We believe that this set of values give us enough values to see the effect of this parameter in altering the empirical distribution of $p(w)$ in order to give rank to words of the document summarized.

Number of Keywords (%)

As a starting point to decide the values of this parameter, we rely on TopicLSA (Melese 2009). We use experiments we conducted on TopicLSA on our own data set. TopicLSA selects a fixed number of terms (3, 5, 6, 8, 10, or 15) from each of the dimensions of the reduced term by concept matrix to form a query. To determine what percentage of unique terms [11] this corresponds to, we use the following calculations:

- If we select J number of terms from a single dimension, then the total number of keywords selected will be $J \times number\ of\ topics$.
- The number of topics depends on the extraction rate and the number of sentences of the document: $number\ of\ topics = number\ of\ sentences \times rate$.
- The percentage of unique terms selected will then be $J \times number\ of\ sentences \times rate \times 100 / number\ of\ unique\ terms$.

We calculated the percentage of unique terms for all the 60 documents and for the three extraction rates as well. For each extraction rate we chose J of the best performing system. This value is 10 for 30% and 25% rates and 6% for the 20% rate. The results we got were quite variable. This is expected because for fixed value of J, small documents will attain higher value for the percentage of unique terms while large documents will attain small values. For this reason we averaged the results for all the documents and for each extraction rate and got the following results:

- 27.73% for 30% extraction rate
- 23.11% for 25% extraction rate
- 11.09% for 20% extraction rate

To have a wider coverage of our parameter, we took 23 as starting point and moved three steps both to larger values and smaller values. This gives us the percentage of keywords to be 11, 14, 17, 20, 23, 26, 29, and 32.

[11] We have observed that TopicLSA might include a term more than once in the query because each LSA dimension is independent. We ignore this issue since we are carrying out a rough estimation.

Similarity Measure

This parameter is used for KITS and KIVSM. For KITS it is used for estimating the similarity between the folded query and the reduced representations of the sentences in the topic simplex. We adopt the widely used similarity measures of the JS divergence (section 2.3.4), KL divergence (section 2.3.4) and cosine (section 2.2.2) (Steyvers and Griffiths 2007; Hennig 2009).

For KIVSM we use the similarity measures of cosine and dot product. We have chosen this similarity measures because they are widely used similarity measures in vector space models.

4.3.2. System Parameters of the Summarization Algorithms

Table 4.4 gives the association of system parameters with the various summarization algorithms. An entry of "variable" indicates that this parameter is experimented for all of its values. An entry of "fixed" means that the parameter is experimented for only some of its values. An entry of "NA" means that the parameter does not exist in the algorithm.

Table 4.4: Association of the summarization algorithms with system parameters.

Parameters	**Algorithm**					
	JWTS	**CWTS**	**JWS**	**CWS**	**KITS**	**KIVSM**
Scoring Method	Variable	Variable	Fixed	Fixed	Fixed	Fixed
β	Variable	Variable	Fixed	Fixed	Fixed	Fixed
Convergence Criterion (%)	Variable	Variable	Fixed	Fixed	Fixed	Fixed
Number of Topics	Variable	Variable	Variable	Variable	Variable	Variable
α	NA	NA	Variable	Variable	Variable	Variable
Number of Keywords (%)	NA	NA	Variable	Variable	Variable	Variable
Similarity Measures (JS, KL, cos, dot)	NA	NA	NA	NA	Variable	Variable
βf	NA	NA	NA	NA	Fixed	NA
Convergence Criterion (Iterations)	NA	NA	NA	NA	Variable	NA

System parameters are fixed in some of the algorithms in order to make the number of experiments carried out manageable in size. For the first two algorithms, JWTS and CWTS, all

of their parameters are experimented for their values. For the rest of the algorithms, the scoring method, β, and convergence criteria (%) are fixed to a maximum of three triples. The reason that these parameters are fixed instead of the others is the nature of their influence in the ranking processes of the algorithms. The purpose of these fixed parameters, together with the number of topics, is to generate the PLSA model over which the algorithms operate on. All of the other parameters operate on this model to give ranks to sentences. Because of this, the parameters that we have made variable have a direct influence on how sentence ranks are generated using each algorithm. We expect that good enough values of the scoring method, β, and convergence criterion (%) give acceptable precision/recall results for the other algorithms. Thus, exploring the parameters that we have made variable gives us insight on the properties of the algorithms more than the fixed parameters give us and at the same time possible high values of precision/recall.

We have made the number of topics variable for all algorithms even though it is used in the construction of the PLSA model. The reason is that the number of topics has a direct influence on the ranking algorithms, as can be seen from the discussions of section 3.6.

As discussed above, we experiment JWTS and CWTS for all values of the scoring method, β, and convergence criteria (%). We then select a maximum of three triples of these parameters from each algorithm that best represent the three extraction rates and use these as fixed parameter values for the rest of the algorithms. Our assumption here is that values of scoring method, β, and convergence criteria (%) that have performed best for any two algorithms are likely to represent the creation of good PLSA models, which can also perform well with the other algorithms.

4.3.3. Experiment 1 – JWTS Algorithm

Goal

This experiment is carried out to assess the performance of JWTS (joint word topic sentence) in terms of precision/recall.

Experimental Setup

Considering the number of values the four parameters above can take, we have a total of $13 \times 9 \times 3 \times 5 = 1755$ system variations. Ideally, a single system variation needs to be evaluated for precision/recall against all the 60 documents of our data set. However, this is a very large number of experiments to do. For this reason, we divide experiment 1 into two different experiments to make the size of the experiment manageable: experiment 1.1 and experiment 1.2. The purpose of experiment 1.1 is to fix the value of scoring method for use in experiment 1.2. We assume that the scoring method does not affect the performance of the algorithm by a significant margin as compared to the other parameters. We will choose a maximum of three scoring methods, each representing an extraction rate. Experiment 1.2 will test the performance of JWTS for the chosen values of the scoring methods and for all combinations of the other parameters (i.e. $,9 \times 3 \times 5 = 135)$. Thus, the performance of JWTS will be judged by the outcome of experiment 1.2.

4.3.3.1. Experiment 1.1

Goal

The goal of this experiment is to fix the value of the scoring method parameter of JWTS to a maximum of three different scoring methods for use in experiment 1.2.

Experimental Setup

As discussed above, we have 13 scoring methods. Ideally, each scoring method needs to be experimented for all 135 combinations of β, convergence criterion and number of topics. Since this approach is not feasible we designed experiment 1.1. Our aim is to choose three representative values from the 135 triples, experiment them with the 13 scoring methods and finally choose three best scoring methods, one for each extraction rate. We choose 2, 4 and 6 for number of topics; 0.7, 0.8, and 0.9 for β. These values are chosen to give a wider coverage for the respective parameters. These values of parameters are then randomly paired. We then randomly pair these with the three values of convergence criterion to get three random triples out of the total of 135:

- $Number\ of\ topics = 2,\ \beta = 0.8, convergence\ criterion = 0.03\%$
- $Number\ of\ topics = 4,\ \beta = 0.7, convergence\ criterion = 0.9\%$

- $Number\ of\ topics = 6,\ \beta = 0.9, convergence\ criterion = 0.007\%$

Each triple is then experimented with all of the 13 scoring methods. When we conducted our experiments, we observed that each run of our experiments gave slightly different results under the same experimental conditions. This is due to the effect of randomized initial conditions of the EM algorithm which converges to a local maximum. To overcome this effect, we run repeated experiments and used the average value as a final result. This ensures the robustness of the scoring methods that will be used for a large number of experiments in experiments 1,3,4,5 and 6.

Results

The full experimental results can be found in Appendix F, table F.1 (table F.1.1, table F.1.2, and table F.1.3). Here, in table 4.5, we present the best performing parameters from each of the triples for each extraction rate.

Table 4.5: Best performing parameters of experiment 1.1.

Scoring Method	β	Number of Topics	Convergence Criterion (%)	Precision/Recall Extraction Rate (%)		
				30	25	20
Bottom 10 of 20 restarts	0.8	2	0.03	**0.48668**	0.45203	**0.42707**
20 restarts				0.48538	**0.45228**	0.41672
15 restarts	0.7	4	0.9	0.50583	**0.4667**	0.4301
Top 15 of 20 restarts				0.49691	0.46052	**0.43149**
Bottom 15 of 20 restarts				**0.50586**	0.46299	0.42175
Bottom 1 of 20 restarts	0.9	6	0.007	**0.45086**	**0.42643**	**0.39982**

Discussion

The bold numbers in each triple have the best performing parameters for that specific triple, while the results marked by bold borders have the best performing parameters from the entire table[12]. For the first triple, the scoring method of bottom 10 of 20 restarts gives the best result for both 30 and 20% rates, while 20 restarts gives the best result for the 25% extraction rate. Unlike

[12] This convention will be used throughout the thesis to differentiate the best performing parameters of a group of experiments from the best performing parameters of the entire experiment.

the first triple, the second triple requires three different scoring methods to represent each of the extraction rates. 15 PLSA models, with different position parameters, give the best results for all extraction rates. The third triple gives the lowest precision/recall result, all extraction rates represented by bottom 1 of 20 restarts. This means that for the third triple, only one PLSA model, which has the lowest log-likelihood out of 20 restarts, is sufficient for summarization. The scoring methods from the second triple ($\beta = 0.7$, number of topics = 4 and convergence criterion (%) = 0.9) give the best results and hence they will be used in experiment 1.2. They are:

- 15 restarts
- Top 15 of 20 restarts
- Bottom 15 of 20 restarts.

4.3.3.2. Experiment 1.2

Goal

The goal of experiment 1.2 is to evaluate the performance of JWTS for precision/recall. As input, it uses three scoring methods from experiment 1.1.

Experimental Setup

Experiment 1.2 is carried out by fixing the scoring method parameter of JWTS to the three best scoring methods from experiment 1.1:

- 15 restarts
- Top 15 of 20 restarts
- Bottom 15 of 20 restarts.

Each of these scoring methods is then experimented for 135 system variations (triples) of the number of topics, β, and convergence criterion. It is to be noted that each system variation is tested for precision/recall performance on all of the 60 documents.

Results

The full experimental results can be found in Appendix F, table F.2. Table 4.6 shows the best performing combinations of parameters of experiment 1.2 for the three scoring methods.

Table 4.6: Best performing combinations of parameters of experiment 1.2 for the three scoring methods.

Scoring Method	β	Number of Topics	Convergence Criterion (%)	Precision/Recall Extraction Rate (%)		
				30	25	20
15 restarts	0.65	3	0.9	**0.51335**	0.45838	0.42968
	0.85	3	0.9	0.49645	**0.47098**	0.42509
	0.9	2	0.9	0.50272	0.46924	**0.4353**
Top 15 of 20 restarts	0.7	2	0.9	**0.51165**	0.46765	**0.43747**
	0.9	3	0.9	0.50103	**0.47274**	0.43092
Bottom 15 of 20 restarts	1	2	0.9	**0.52012**	0.46791	0.42549
	0.7	2	0.007	0.494	**0.47281**	0.43066
	0.6	2	0.3	0.49742	0.45539	**0.44068**

As discussed in section 4.3.2, there are four algorithms that need to have a fixed value for their scoring method, β, and convergence criterion parameters. We need to choose a maximum of three triples of these parameters for use in the other algorithms. One way of doing this is to pick those triples from table 4.6 that have the best precision/recall, from each extraction rate. This results in the choice of all the parameters corresponding to the scoring method of bottom 15 of 20 restarts. But if we look at the full results of table F.2.3, we see that each of these parameters favor only a certain number of topics, especially 2 number of topics. But the various algorithms for which we fix these parameters for may work best on other number of topics. For this reason, we remove the effect of the number of topics on the precision/recall scores of the triples of parameters of table F.2 by assigning the average of the precision/recall values corresponding to the 5 number of topics, for each extraction rate. This averaging is done for all of the tables represented by table F.2.

For example, taking table F.2.1, let us calculate the score for the triple

- Scoring Method = 15 restarts
- $\beta = 1$
- Convergence Criterion = 0.9.

This triple has a precision/recall score corresponding to each of the five number of topics and three of the extraction rates. If we take the 30% extraction rate, the precision/recall values are: 0.50768, 0.47579, 0.45332, 0.46301, and 0.44444. These five values are then averaged to give 0.46885. This value becomes the score of our example triple at 30% extraction rate. The full table for each scoring method is shown in Appendix F, table F.3. Here, we present table 4.7 that shows the best triples selected from each scoring method, representing each extraction rate.

Table 4.7: Best values for the averaged table (across topics) of F.2.

Scoring Method	**β**	**Convergence Criterion (%)**	**Precision/Recall**		
			Extraction Rate (%)		
			30	**25**	**20**
15 restarts	0.65	0.9	**0.50879**	**0.45938**	0.42441
	0.6	0.9	0.50844	0.45718	**0.42788**
Top 15 of 20 restarts	0.7	0.9	0.49759	**0.46111**	**0.43024**
	0.6	0.9	**0.50862**	0.45916	0.42848
Bottom 15 of 20 restarts	0.7	0.9	0.49812	**0.4604**	0.4252
	0.6	0.9	**0.51005**	0.45661	**0.42775**

Discussion

We can see from the results of the performance table, table 4.6, that the scoring method of bottom 15 of 20 restarts is associated with all of the best systems, showing that it is favored by JWTS than other scoring methods. The experiments show that two and three number of topics give the best performing systems. The same, near homogenous trend, applies for the convergence criterion. The convergence criterion of 0.9 gives the best results almost in all cases, indicating that the PLSA model becomes good enough without much training (usually 5 to 20 number of iterations). This makes the EM algorithm much faster. More generally, we can see from table F.2 that increasing the number of iterations for the EM algorithm (i.e., decreasing the convergence criterion (%)) results in reduced performance. Our experiments do not give a clear value of β that can be labeled the best. Table 4.6 shows that the value of β for the best performing combinations of parameters depends on the other parameters.

The results shown in table 4.7 show us which scoring method and β pairs are stable across the number of topics. The table shows that the best pairs have performance that is one less than the

best combinations of parameters of table 4.6. Table 4.7 shows that the scoring method of top 15 of 20 restarts performs well for low extraction rates while the scoring method of bottom 15 of 20 restarts performs well for 30% extraction rates. On average, we see from the table that, the convergence criterion of 0.9 gives the best stable scores across topics, showing that high quality PLSA models are achievable at small number of iterations. Table 4.7 also confirms our observation that the best value of β depends on other parameters. In this case a higher tempering parameter (0.7) is associated with top 15 of 20 restarts, while a lower tempering parameter of 0.6 is associated with bottom 15 of 20 restarts. The table also shows that lower values of β are not affected as much as higher values when the number of topics is varied.

4.3.4. Experiment 2 – CWTS Algorithm

Goal

The goal of experiment 2 is to evaluate the performance of the summarization algorithm CWTS for precision/recall.

Experimental Setup

The experimental setup of experiment 2 mirrors that of the previous experiment, experiment 1. We could have used the scoring method, β, and convergence criteria values from experiment 1, table 4.7. But, the normalization that CWTS uses for ranking could make the algorithms favor a different set of parameters. Because of this, we will carry out the exact experimental setup used in experiment 1 so that we can see which parameter values are favored by CWTS.

Experiment 2 is divided into two different experiments: experiment 2.1 and experiment 2.2. Experiment 2.1 will be used to decide a maximum of three scoring methods, each representing the three extraction rates. These values will be given as input to experiment 2.2. Experiment 2.2 will use each scoring method from experiment 2.1 to test the performance of CWTS for the 135 triples of the number of topics, convergence criterion and β.

4.3.4.1. Experiment 2.1

Goal

This experiment is carried out to decide the value of the scoring method parameter of CWTS to a maximum of three values for use in experiment 2.2. Each scoring method represents the best precision/recall value for each extraction rate.

Experimental Setup

Here, we decide which of the 13 scoring methods available are best for CWTS. We use the same random triples out of the 135 values of the parameters number of topics, β, and convergence criterion as in experiment 1.1. Each random triple is then used with each of the 13 scoring methods to generate summaries for the 60 documents. Summarization is repeated in all cases and the average precision/recall result is recorded.

Results

The outcome of experiment 2.1 produces the exact kind and number of tables found in Appendix F, table F.1. For this reason, we report only the best performing parameters from each of the triples representing the best performing parameters for each extraction rate. The results are shown in table 4.8 below.

Table 4.8: Best performing parameters of experiment 2.1

Scoring Method	β	Number of Topics	Convergence Criteria (%)	Precision/Recall Extraction Rate(%)		
				30	25	20
Bottom 5 of 20 restarts	0.8	2	0.03	**0.40584**	**0.3827**	**0.36432**
Bottom 10 of 20 restarts	0.7	4	0.9	0.42554	**0.40241**	0.37576
20 restarts				0.42473	0.4002	**0.38376**
1 random start	0.9	6	0.007	0.42668	0.39503	**0.37978**
Bottom 1 of 20 restarts				**0.43691**	**0.3977**	0.37334

Discussion

For the first triple, the scoring method of bottom 5 of 20 restarts gives the best result for all extraction rates. This is an interesting stable performance with respect to a scoring method which

was also observed in experiment 1.1, table 4.5, for the scoring method of bottom 1 of 20 restarts. From table 4.8 we see that the best scoring methods that will be used for experiment 2.2 are bottom 10 of 20 restarts (representing the best system at 25% extraction rate), 20 restarts (representing 20% extraction rate) and bottom 1 of 20 restarts (representing 30% extraction rate).

Comparing this table with table 4.5 of algorithm JWTS shows that, CWTS seems a "poor" algorithm, with lower precision/recall scores (although the remaining 147 parameter triples are yet to be explored). Table 4.8 shows us that most of the scoring methods use a small number of PLSA models (1, 5 and 10 – 10 may be considered medium). We may cautiously conclude that a small number of PLSA models perform well on poor summarization parameters/algorithms than large number of PLSA models. This is of course in a direct contrast to the results of table 4.5 where large number of PLSA models (15 and 20) performed the best on parameters/algorithm that have high precision/recall value. Table 4.5 also supports our conclusion since the triple of parameters represented by the last row performs poorly and this is achieved by the scoring method of bottom 1 of 20 restarts (the reader can refer to table F.1.3 in the appendix for more details on the data used to arrive at this conclusion).

4.3.4.2. Experiment 2.2

Goal

The aim of experiment 2.2 is to evaluate the performance of CWTS for precision/recall. It accepts three values for the parameter of scoring method from experiment 2.1.

Experimental Setup

The performance of CWTS is evaluated by carrying out experiments where its scoring method parameter is fixed to those identified as best in experiment 2.1. Then, each of the three chosen scoring method is experimented for the 135 triples of the rest of the parameters (number of topics, β, and convergence criterion). The chosen scoring method from experiment 2.1 are:

- Bottom 1 of 20 restarts.
- Bottom 10 of 20 restarts.
- 20 restarts.

Results

The result of experiment 2.2 produces tables that are similar to those of experiment 2.1. Table 4.9 below shows the best performing combinations of parameters for each of the three scoring methods of experiment 2.2.

Table 4.9: Best performing combinations of parameters of experiment 2.2 for the three scoring methods.

Scoring Method	β	Number of Topics	Convergence Criterion (%)	Precision/Recall Extraction Rate(%)		
				30	25	20
Bottom 1 of 20 restarts	1	6	0.9	**0.44809**	0.41214	0.3738
	0.8	6	0.03	0.43738	**0.417**	0.37773
	0.95	6	0.007	0.42788	0.40703	**0.40285**
Bottom 10 of 20 restarts	0.65	3	0.9	**0.45044**	**0.4216**	**0.39367**
20 restarts	0.95	6	0.9	**0.43461**	**0.40951**	0.37766
	0.65	6	0.9	0.43062	0.40326	**0.3872**

We have one remaining table to report. This table is a table similar to table 4.7 of experiment 1. The purpose is to choose a maximum of three scoring method, β, and convergence criteria triples for use in the algorithms of JWS, CWS, KITS and KIVSM. To remove the influence of the number of topics, we average the precision/recall value of a triple, independently for each extraction rate. The resulting table looks like table F.3 of experiment 1. Table 4.10 below shows only the best results from each scoring method.

Table 4.10: Best values for the averaged table (across topics) of the full table of experiment 2.2.

Scoring Method	β	Convergence Criterion (%)	Precision/Recall Extraction Rate(%)		
			30	25	20
Bottom 1 of 20 restarts	1	0.03	**0.42512**	0.39089	0.36678
	0.65	0.007	0.42413	**0.39808**	0.37371
	0.75	0.03	0.42144	0.39756	**0.37704**
Bottom 10 of 20 restarts	0.65	0.9	**0.43143**	**0.40432**	**0.37722**
20 restarts	0.65	0.9	**0.42592**	**0.39607**	0.37418
	0.75	0.9	0.41756	0.39551	**0.37601**

Discussion

Table 4.9 shows that the performance of CWTS is significantly lower than that of JWTS. The performance of CWTS is also lower than PROC3+. CWTS normalizes sentence length before counting the number of keywords contained in a sentence. JWTS does not normalize sentence length and PROC3+ simply selects the longest sentences. This suggests that normalizing sentence length for text summarization is not good for the keyword-based summarization algorithms that we have proposed.

The best systems are for the scoring methods of bottom 1 of 20 restarts and bottom 10 of 20 restarts, supporting our previous conclusion that small number of PLSA models perform well on low performing systems. We also observe that top based scoring methods, except for top 15 of 20 restarts, perform poorly both in JWTS and CWTS.

Contrary to JWTS, CWTS works best under a large number of topics, six for most of the best systems. But three number of topics gives the best result for 30% and 25% extraction rates. We recall that two number of topics gave the best results for JWTS.

As for the parameter of the convergence criterion we see two distinct patterns. Bottom 1 of 20 restarts does best when we have more number of iterations (convergence criteria is either 0.03% or 0.007%). But the other scoring methods, which use many PLSA models for sentence scoring, prefer a convergence criterion of 0.9%, just like JWTS. Similar to the case of JWTS, table 4.9 shows that the optimum value of β depends on other parameters and a distinct value of β is not favored.

Table 4.10 shows that only a single triple is required to represent all extraction rates; bottom 10 of 20 restarts, $\beta = 0.65$ and convergence criterion 0.9. We again observe from table 4.10 that bottom 1 of 20 restarts prefers large numbers of iterations to get large values of precision/recall, while the other scoring methods need only a small number of iterations.

Table 4.11 below summarizes the triples of the parameters scoring method, β, and convergence criterion that are chosen from experiment 1.2 (table 4.7) and experiment 2.2 (table 4.10).

Table 4.11: Values of scoring method, β, and convergence criterion chosen for subsequent algorithms (JWS, CWS, KITS and KIVSM).

Scoring Method	β	Convergence Criterion (%)	Chosen from Experiment	Chosen for Extraction Rate(s) of (%)
Top 15 of 20 restarts	0.7	0.9	1.2	25 and 20
Bottom 10 of 20 restarts	0.65	0.9	2.2	30, 25 and 20
Bottom 15 of 20 restarts	0.6	0.9	1.2	30

We see from table 4.11 that each extraction rate is represented twice. The convergence criterion is 0.9% for all triples, which will also make the running time of the algorithms much faster. We also observe that different values of β are associated with different scoring methods.

4.3.5. Experiment 3 – JWS Algorithm

Goal

The goal of experiment 3 is to evaluate the performance of the JWS (joint word sentence) summarization system for precision/recall.

Experimental Setup

Experiment 3 is carried out by fixing the scoring method, β, and convergence criterion parameters of JWS to values given in table 4.11. The other parameters of JWS are then tested with each triple of values of table 4.11. The number of system variations due to the parameters that have not been fixed (number of keywords, number of topics and α) is $8 \times 5 \times 4 = 160$.

Results

The full experimental results can be found in Appendix F, table F.4. Table 4.12 shows the best performing combinations of parameters of experiment 3 for the three scoring methods and the other two fixed parameters.

Table 4.12: Best performing combinations of parameters of experiment 3 for the three scoring methods.

Scoring Method, β, Convergence Criterion (%)	Number of Keywords (%)	Number of Topics	α	Precision/Recall Extraction Rate (%)		
				30	25	20
Top 15 of 20 restarts, 0.7, 0.9	23	3	5	**0.51644**	0.46852	0.43963
	26	6	4	0.50324	**0.48499**	0.45472
	29	6	4	0.48942	0.46179	**0.45511**
Bottom 10 of 20 restarts, 0.65, 0.9	20	2	5	**0.51669**	0.45992	0.41828
	23	6	5	0.50714	**0.4656**	0.42702
	11	6	3	0.50831	0.46385	**0.43623**
Bottom 15 of 20 restarts, 0.6, 0.9	23	2	3	**0.516**	0.46295	0.43055
	17	3	5	0.50938	**0.46403**	0.42644
	20	6	5	0.51067	0.45726	**0.43141**

Discussion

Table 4.12 shows that JWS has performance values similar to JWTS. This is because of the similarity between their sentence ranking algorithms. That is, both use joint probability rather than conditional probability to estimate the presence of keywords in a sentence.

The results show that the scoring method of top 15 of 20 restarts gives the best systems for the low extraction rates of 25% and 20%, while bottom 10 of 20 restarts performs best at the higher extraction rate of 30%. The table also shows that in most cases, the best systems favor a high value of α (4 and 5). This means that the dominant topics have greater influence in the term ranking process. We can also observe that a large percentage of unique words are usually required to compose the best summaries. A particular number of topics do not seem to be favored; the best value for the number of topics seems to depend on the values of the other parameters.

4.3.6. Experiment 4 – CWS Algorithm

Goal

The purpose of this experiment is to evaluate the performance of the CWS (conditional word sentence) summarization algorithm for precision/recall.

Experimental Setup

The experimental setup of experiment 4 is similar to that of Experiment 3 because of the similarity between the summarization algorithms CWS and JWS. The scoring method, β, and convergence criterion parameters of CWS are fixed with values obtained from experiments 1 and 2 (table 4.11). The combination of parameters that are allowed to vary in experiment 4 (number of keywords, number of topics and α) have a total of 160 triples. Each of these triples is tested with the three fixed parameter sets of table 4.11.

Results

Experiment 4 results in tables similar to that of table F.4. We report only the best performing combinations of parameters from each of the three scoring methods. The results are given in table 4.13 below.

Table 4.13: Best performing combinations of parameters of experiment 4 for the three scoring methods.

Scoring Method, β, Convergence Criteria (%)	Number of Keywords (%)	Number of Topics	α	Precision/Recall Extraction Rate (%)		
				30	25	20
Top 15 of 20 restarts, 0.7, 0.9	32	6	3	**0.45139**	0.4155	0.39228
	32	5	4	0.44631	**0.42393**	0.40016
	20	5	2	0.43738	0.41349	**0.41327**
Bottom 10 of 20 restarts, 0.65, 0.9	32	5	2	**0.44773**	0.39752	0.38215
	23	6	5	0.4306	**0.41613**	0.39188
	23	5	3	0.44017	0.41257	**0.39533**
Bottom 15 of 20 restarts, 0.6, 0.9	32	5	2	**0.44698**	0.41353	0.37808
	17	4	4	0.44460	**0.41432**	0.39874
	29	5	3	0.43547	0.40801	**0.40327**

Discussion

We can see from the results of experiment 4 that CWS has a low performance like that of CWTS. This is because both use conditional probabilities to identify keywords in sentences, which is a form of sentence length normalization.

The results show that the best systems are all attainable with the scoring method of top 15 of 20 restarts. This is an improved performance by this scoring method from the previous experiment (experiment 3) in that it gives the best systems for all extraction rates. As in the case of JWS, a large percentage of unique words give the best summarization systems.

A trend that is different from that of JWS is the behavior of the parameter used in term ranking, α, and the number of topics. CWS seems to require a large number of topics to produce good summaries. In contrast to this, α for the best systems is seen to depend on the values of the other parameters.

4.3.7. Experiment 5 – KITS Algorithm

Goal

The purpose of this experiment is to evaluate the performance of KITS (keywords in topic simplex) for precision/recall.

Experimental Setup

The parameters of KITS is the same as JWS and CWS except for the parameters similarity measure, βf, and convergence criterion (iterations). The addition of these parameters forces us to adopt an experimental setup similar to JWTS and CWTS. The similarity measure parameter does not create a problem since it has only three values; JS (Jensen-Shannon), KL (Kullback Leibler) and cosine. That is we can run experiment for all of these values. βf has been fixed to the value of 1 as discussed in section 4.3.1. We, therefore, remain with the parameter convergence criterion (iterations). Thus, we have four variable parameters as in the case of JWTS and CWTS. This means that we need a preliminary experiment to fix one of the four variable parameters so that we have a manageable number of experiments. From the four variable parameters (number of keywords, α, number of topics, and convergence criterion (iterations)), we have decided to fix the convergence criterion (iterations). The reason for this is that the other parameters have a more direct influence on the behavior of KITS than the convergence criterion as can be seen from the discussion of section 3.6.3.1.

Once we have decided to fix the value of convergence criterion (%), the next issue is how many values we need to fix, possibly each representing the three extraction rates. In the case of JWTS/CWTS where the parameter to fix was the scoring method, we fixed three values. But the case of convergence criterion (iterations) is different. We decided to fix only one value of convergence criterion (iterations) based on the following argument. From the results for the performances of JWTS and CWTS we see that the best systems share a common value of convergence criterion (%) which is 0.9%. This leads us to the conclusion that we can associate one convergence value for the creation of a PLSA model to suit all algorithms. That is, that particular value of convergence criterion (%) is an indication of a good PLSA model. Following

this observation we can argue that only one value of convergence criterion (iterations) suffices to produce a good folded query in a PLSA model.

Experiment 5 will then consist of two experiments: experiment 5.1 and experiment 5.2. The purpose of experiment 5.1 will be to fix the convergence criterion (iterations) to one value. Then, experiment 5.2 will be used to evaluate the performance of KITS.

4.3.7.1. Experiment 5.1

Goal

The goal of this experiment is to fix the value of the convergence criterion (iterations) parameter of KITS to one value for use in experiment 5.2.

Experimental Setup

We follow the same experimental setup as in experiments 1.1 and 1.2. We choose three random triples out of the 160 possible values for the parameters number of topics, α, and number of keywords:

- $Number\ of\ topics = 2,\ \alpha = 4, number\ of\ keywords = 17\%$
- $Number\ of\ topics = 4,\ \alpha = 5, number\ of\ keywords = 32\%$
- $Number\ of\ topics = 6,\ \alpha = 2, number\ of\ keywords = 26\%$

Then, these three triples are each experimented for all of the values of the parameters of convergence criterion (iterations). Experiment 5 also uses the three scoring methods from table 4.11. For experiment 5.1, we use top 15 of 20 restarts because it performed best in most cases of the previous algorithms. Each experiment is run twice and the average is reported.

Results

Experiment 5.1 results in three tables that are similar to table F.1. We report only the best performing parameters representing each extraction rate. The results are shown in table 4.14 below.

Table 4.14: Best performing of parameters of experiment 5.1 (Scoring method = Top 15 of 20 restarts, $\beta = 0.7$, convergence criterion = 0.9%).

Convergence Criterion (Number of iterations)	Number of Keywords (%)	Number of Topics	α	Precision/Recall Extraction Rate (%)			Average of a Row
				30	25	20	
20	17	2	4	0.41713	0.38598	**0.38349**	0.39554
100				**0.42623**	**0.39123**	0.37503	0.39750
60	32	4	5	**0.44398**	**0.413**	**0.39698**	**0.41799**
20	26	6	2	0.44637	**0.41783**	0.38488	0.41636
100				0.4392	0.41329	**0.39341**	0.41530
200				**0.44666**	0.4065	0.38678	0.41331

Discussion

We can see from table 4.14 that three different iterations (20, 60 and 200) represent the three extraction rates. But because we have to use only one value for the convergence criterion (iterations), we need to break the tie. To achieve this, we consider average of the performances of each convergence criterion (iterations) across all extraction rates. This is represented in table 4.14 by the column designated by "Average of a Row". From the table we see that 60 number of iterations gives the best results and it will be used in experiment 5.2. This number of iterations also gives the best result for the 20% extraction rate and compares quite well on other extraction rates with the other best system variations (20 and 200 iterations). From our experiments, we conclude that points of overtraining, such as 200, perform well on summarization tasks with respect to the folding operation.

4.3.7.2. Experiment 5.2

Goal

The goal of experiment 5.2 is to evaluate the performance of KITS for precision/recall.

Experimental Setup

The scoring method, β, and convergence criteria parameters of KITS are fixed to values given in table 4.11. The convergence criterion (iterations) parameter is fixed to the value of 60 as discussed in experiment 5.1. We are then left with four parameters that can be varied. As in experiments 3 and 4, we form 160 triples of the parameters number of keywords, number of topics and α. Each of the 160 triples will then be experimented for the three values of the similarity measure parameter (JS, KL and cos). The resulting configuration is also tested for the three scoring methods of table 4.11.

Results

The outcome of experiment 5.2 results in three tables which look like table F.4 of experiment 3. Each of the three tables is associated with exactly one similarity measure (JS, KL or cos). Each similarity measure uses three scoring methods to test the performance of KITS with the 160 triples of the parameters number of keywords, number of topics and α. Table 4.15 below shows the best performing combinations of parameters of experiment 5.2. We have reported only the best systems from each similarity measure parameter that represent the three extraction rates. Because of space constraints, we have left out the values of β, and convergence criteria and presented only the value of the scoring method.

Table 4.15: Best performing combinations of parameters of experiment 5.2 for the three similarity measures of JS, KL and cos.

Similarity Measure	Scoring Method, β, Convergence Criteria (%)	Number of Keywords (%)	Number of Topics	α	Precision/Recall Extraction Rate (%)		
					30	25	20
JS	Bottom 10 of 20 restarts	17	4	3	**0.46069**	0.41649	0.38098
		26	5	4	0.45015	**0.42461**	0.39443
	Top 15 of 20 restarts	26	5	5	0.44226	0.41554	**0.41026**
KL	Top 15 of 20 restarts	32	6	5	**0.46537**	0.43633	0.41513
		17	5	4	0.44964	**0.43709**	0.40537
		23	6	3	0.4541	0.42164	**0.41931**
cos	Bottom 15 of 20 restarts	26	4	2	**0.46655**	**0.4327**	**0.41629**

Discussion

The table shows us that in general the KL similarity measure gives the best performance, followed by cosine and then JS. We are surprised that the cosine similarity measure outperformed the JS similarity measure. The KL and JS similarity measures are specifically created to assess similarity measures between probability distributions, unlike the cosine measure which is supposed to work best in vector space models.

The scoring method of top 15 of 20 restarts again seems to perform better than the other scoring methods. An interesting observation that can be made from the table above is that, the similarity measures of KL and cosine give the best performing systems using only one scoring method. As in the case of JWS and CWS, a large number of keywords is required to produce the best extractive summaries. The required number of topics and the value of α is also high.

4.3.8. Experiment 6 – KIVSM Algorithm

Goal

The goal of experiment 6 is to evaluate the performance of the KIVSM (keywords in vector space model) summarization system for precision/recall.

Experimental Setup

The experimental setup for KIVSM is similar to that of experiments 3 and 4. KIVSM has the same number of parameters like JWS/CWS expect for the similarity measure parameter (cosine and dot product). Thus, the scoring method, β, and convergence criteria parameters of KIVSM are fixed to values given in table 4.11. The other parameters of KIVSM disregarding the similarity measure form 160 triples of values. Each similarity measure is then run with the 160 triples of values and the three scoring methods.

Results

Experiment 6 results in two tables like that of table F.4. Each table represents either the cosine or dot produce similarity measure. Each similarity measure in turn hosts three scoring methods and 160 triples of values (for the parameters number of keywords, number of topics and α) associated with each scoring method. The full experimental results can be found in Appendix F, table F.4. Table 4.16 shows the best performing combinations of parameters of experiment 6 from each of the two similarity measures.

Table 4.16: Best performing combinations of parameters of experiment 6 for the two similarity measures of cos and dot product.

Similarity Measure	Scoring Method, β, Convergence Criteria (%)	Number of Keywords (%)	Number of Topics	α	Precision/Recall Extraction Rate (%)		
					30	25	20
cos	Bottom 15 of 20 restarts	26	3	5	**0.50521**	0.45673	0.42207
		29	2	2	0.492	**0.46465**	0.4243
	Top 15 of 20 restarts	29	5	5	0.48204	0.45483	**0.43743**
dot	Top 15 of 20 restarts	32	2	2	**0.51721**	0.46862	0.42527
	Bottom 15 of 20 restarts	32	4	4	0.50438	**0.48076**	0.41928
	Top 15 of 20 restarts	29	6	4	0.5027	0.46298	**0.44816**

Discussion

We observe from the results that the dot product similarity measure gives the best precision/recall results in most cases. This is in line with our previous observation that sentence length normalization reduces performance. In the case of CWTS and CWS the effect is severe. But we are surprised that the effect is not that severe in the case of KIVSM-cos (KIVSM with the

cosine similarity measure). As in the previous cases, a high number of keywords are required to compose the best extractive summaries. As can be seen from the table, no particular values of the number of topics and α are favored by KIVSM.

We have observed an interesting behavior of KIVSM with respect to the parameter of number of keywords. The observation holds for all experiments done concerning KIVSM. For illustration purposes, we present precision/recall results that contain the best system variation at 30% extraction rate of KIVSM-dot (KIVSM used with the dot product) at $\alpha = 2$. See figure 4.3 below.

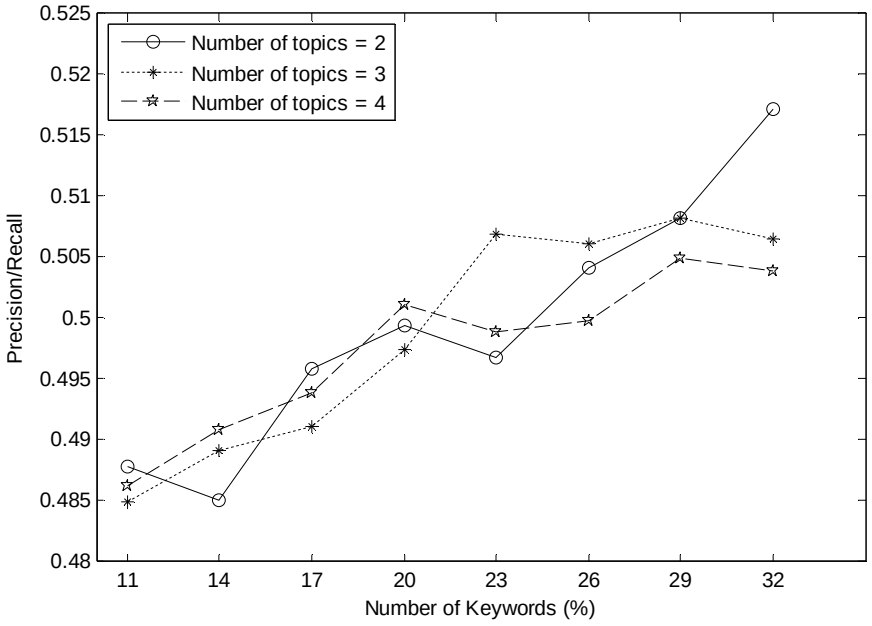

Figure 4.3: Performance variations of KIVSM-dot that contains the best system variation at 30% extraction rate, at $\alpha = 2$.

As shown in figure 4.3, the performance of KIVSM-dot tends to increase steadily when the number of keywords used to rank sentences increases. Lower values of the number of keywords are particularly associated with low values of precision/recall results. This indicates that the half-concept-based approach, KIVSM, needs a large number of important point indicative keywords to create good summaries. We did not observe such a pattern in JWS, CWS or KITS. The later algorithms, although they perform better at higher values of the number of keywords, also

perform very well at lower values of the number of keywords. This is the case because KIVSM uses the original sparse term by sentence matrix to detect the presence of keywords in sentences. A keyword that has been selected at the word ranking stage is thus forced to have influence only in the sentences that it originally appeared in when used in the sparse term by sentence matrix for sentence ranking. However, in the case of JWS, CWS and KITS, they use the reliable representations of the document to be summarized that result from the PLSA model ($p(w,s)$ and $p(w|s)$). These representations are not sparse and hence during sentence ranking, a certain keyword will have influence not only in the original sentences that it appeared but also in other sentences that are conceptually related to it as determined by the PLSA model. This enables the algorithms that use a full-concept-based approach to be robust to low values of the number of keywords parameter.

4.3.9. Experiment 7 – Summarization Algorithms without the First Sentence

Table 4.2 of section 4.2.3 has shown that the inclusion of the first sentence in the summarization of PROC3 has a significant effect on the resulting algorithm of PROC3+. Considering the nature of the documents we experimented with (news articles), this is also expected for our proposed algorithms. As a confirmation of this behavior, we have tested the performance of one of our best proposed systems, JWS, by avoiding the inclusion of the first sentence during summary generation. The resulting algorithm is labeled JWS-. The result is shown in table 4.17 below.

Table 4.17: Performance of JWS without first sentence.

Summarizer	Precision/Recall		
	Extraction Rate (%)		
	30	25	20
JWS	**0.50324**	**0.48499**	**0.45472**
JWS-	0.45469	0.41362	0.36612
PROC3	0.44956	0.39045	0.30935

The parameters used to generate summary for JWS- are that of the best performing JWS system variation at 25% extraction rate (table 4.12). The performance of JWS shown is also taken from table 4.12. This system variation of JWS differs from JWS- only by the inclusion of the first sentence while the summary is created.

The results show that JWS- performs well below JWS as expected. However, JWS- performs better than PROC3. The difference in performance between JWS- and PROC3 becomes more significant at the 20% extraction rate.

4.4. Discussion

In this chapter, we evaluated the performance of the proposed summarization algorithms. Precision/recall measurements were used for the evaluation purpose at different rates of extraction; 20%, 25% and 30%.

Our experimental results show that the inclusion of the first sentence of the document to the summary has shown to be very effective in taking advantage of the genre information of the summarized documents, which are news reports.

The various parameters of the summarization system play a crucial role in helping algorithms achieve their full potential. In particular, a small number of iterations of the EM algorithm that generates the PLSA model is found to be quite effective in creating high quality PLSA models. Increasing the number of iterations further produces in significant reduction in precision/recall scores. Lower values of the tempering parameter β(0.6, 0.65 and 0.7), have also produced quite stable and good performance. The scoring method of top 15 of 20 restarts coupled with $\beta = 0.7$ has performed quite well in most algorithms, especially at lower extraction rates. This shows that the problem of local maxima that affect the EM algorithm can be tackled well consistently by a single scoring method. This is an improvement over previous proposed approaches that use the method of 5 restarts to solve the same problem (Brants 2002; Hennig 2009). We have also found out that large numbers of keywords are required in all algorithms to produce the best summaries.

Previous experiments in information retrieval (Deerwester et al. 1990; Hofmann 2001) have shown quite clearly that topic modeling using statistical methods via co-occurrence analysis of a text without the use of other knowledge sources can detect conceptual relations between words. Our experiments have indirectly shown us that this is the case. More specifically, we implemented the half-concept-based approach algorithm of KIVSM. We found out that this half-

concept-based approach always needs a large number of keywords to achieve its full potential, with its performance increasing steadily as the number of keywords is increased. However, the full-concept-based approaches that make explicit use of keywords (JWS, CWS and KITS) can perform near their best with a small number of keywords. In some cases, the use of small number of keywords turns out to be the best method depending on the scoring method used. This means that the latter algorithms are able to use a small number of keywords to discover most of the salient points of the text, which would not be possible if term matching is used to select summary sentences using a small number of keywords.

We have made a comparison of the performances of the proposed algorithms with one another and the other systems that we have introduced earlier. We take two of the best systems that have been introduced in section 4.2.3 for comparison with our proposed algorithms: PROC3+ and First *n* Sentence. The comparisons are made separately for each extraction rate.

Comparison at 30% Extraction Rate

Figure 4.4 below shows the precision/recall scores of the best of the proposed systems and the compared systems at 30% extraction rate.

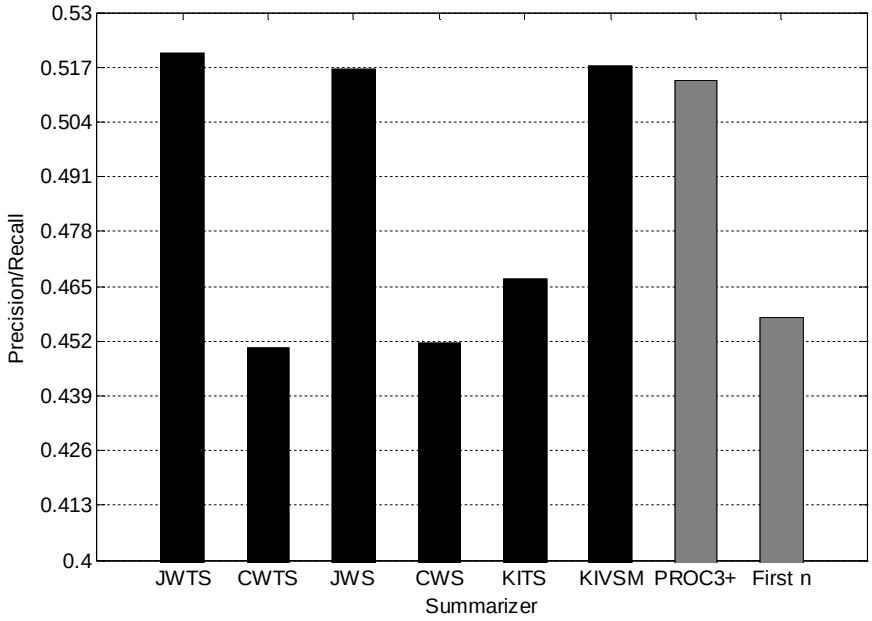

Figure 4.4: Best of proposed systems and compared systems at 30% extraction rate.

While JWTS is the best performing system at 30% rate, it can be seen that systems that have the best performance are the algorithms that favor long sentences; they are JWTS, JWS, KIVSM-dot and PROC3+. This can be explained by the argument that the longer the sentence, the higher the chance that it contains useful concepts. PLSA gives a formal proof that the longest sentences are the likely candidates for inclusion in summary. This has been shown by (Bhandari et al. 2008) and proved to be correct in their experiment on a standard English data set of the DUC 2002. Bhandari et al. (Bhandari et al. 2008) implemented the longest sentences algorithm (PROC3) and it outperformed all the compared systems (such as HITS and LexPageRank). The principle of PROC3 is to pick those sentences that score high with respect to most topics. This can be achieved by having the sum $\sum_z p(s|z)p(z)$. What this sum does is that it weights the score of a sentence with respect to a topic, $p(s|z)$, by the importance (magnitude) of the topic, $p(z)$, and

sums the contribution from all topics. But we have $p(s) = \sum_z p(s|z)p(z)$, where $p(s)$ is simply the length of a sentence in words.

Algorithms CWTS, CWS and KITS use some form of normalization of a sentence feature with respect to the sentence length. This resulted in decreased precision/recall performance. But the performance of KITS is higher, followed by CWS and then by CWTS. KITS seems less prone to the normalization issue, probably because it takes into account all words of the document to give sentence scores. The scoring algorithm of CWTS cancels out the effect of the keywords in the final sentence scores. It is for this reason, we think that CWS, which takes into account the contribution from keywords, slightly does better than CWTS. But both CWTS and CWS are outperformed by first *n* sentences algorithm.

As shown in figure 4.4, all of the proposed algorithms that favor longest sentences perform better than the modified algorithm of (Bhandari et al. 2008). This shows us that the longest sentences algorithm can benefit from the inclusion of shorter sentences once in a while. But to be precise, we have not made empirical study on the proportion of the longest sentences selected by the proposed approaches that favor long sentences. It is surprising that the half-concept-based approach KIVSM-dot slightly outperforms JWS. This shows that the vector space model can have good performance if combined with concept-based techniques, in this case the keyword selection process. In contrast to the case of CWTS and CWS, JWTS which cancels the effect of the keywords in the ranking process outperforms JWS. It can be seen from figure 4.4 that the first *n* sentences performs poorly with respect to the algorithms that favor long sentences.

Comparison at 25% Extraction Rate

Figure 4.5 below shows the precision/recall scores of the best of the proposed systems and the compared systems at 25% extraction rate.

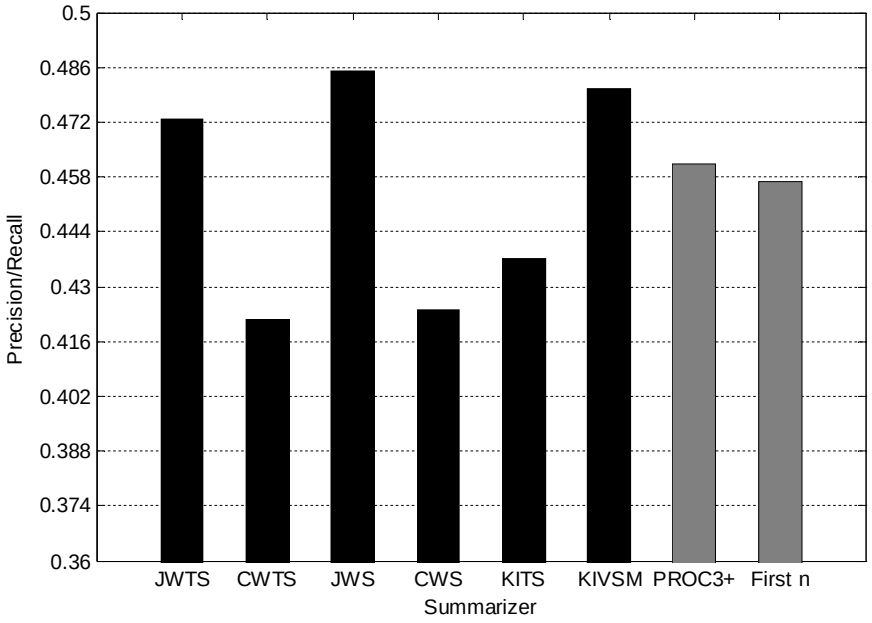

Figure 4.5: Best of proposed systems and compared systems at 25% extraction rate.

The situation at 25% extraction rates is largely the same as that of at 30% except for a few differences:

- JWS now performs much better than JWTS, as expected from the fact that JWTS ignores the influence of the selected keywords.
- KIVSM-dot now performs better than JWTS. This means that the keywords counting procedure of KIVSM-dot[13] compensates for its use of a half-concept-based approach.
- JWS also outperforms KIVSM-dot. The use of the concept-based sentence ranking procedure of JWS seems too much for KIVSM-dot at lower and hence more demanding extraction rate. Certainly, we expect that more "thinking" is needed to produce shorter summaries than longer ones.
- PROC3+ now gets outperformed more significantly than previously by all of the proposed approaches that favor long sentences. This is expected because as the extraction rate goes lower, the chance that a longer sentence gets selected decreases. Hence, more analysis is required to get better precision/recall results than just selecting the longest sentences.

[13] Keyword counting because it uses the dot-product as a similarity measure and sentence scores are always integers.

- The first *n* sentences summarizer now performs better than KITS and it has now performance that is comparable to the algorithms that favor long sentences. But it is still outperformed by the algorithm that uses the longest sentences heuristic.

Comparison at 20% Extraction Rate

Figure 4.6 below shows the precision/recall scores of the best of the proposed systems and the compared systems at 20% extraction rate.

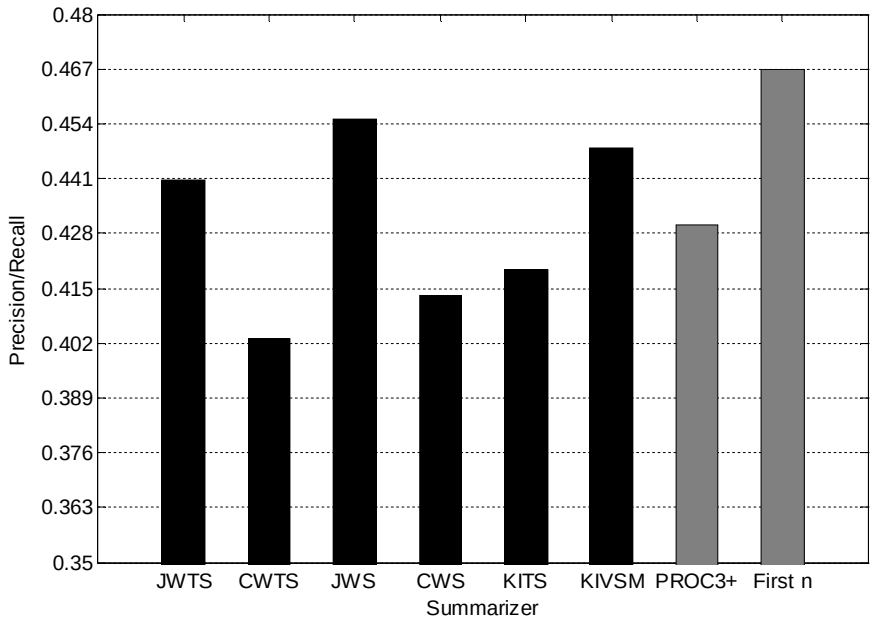

Figure 4.6: Best of proposed systems and compared systems at 20% extraction rate

The way the different algorithms compare at 20% extraction rate mirrors the one at 25% rate, the differences being:

- The first *n* sentences algorithm now performs better than all of the algorithms. This confirms previous results in the literature that this heuristic is a strong baseline for text summarization in the news domain.
- CWS has now performance better than CWTS by a bigger margin than at 30% and 25% extraction rates. This shows that the lack of use of keywords by CTWS affects its performance strongly at lower extraction rates.

We have observed that the keyword approach using LSA by (Melese 2009), TopicLSA, has performance slightly below CWTS and well outperformed by the best proposed systems that favor long sentences. The reason for this can be explained by the fact that TopicLSA uses the cosine similarity measure to rank sentences, which has the effect of normalizing sentence length. However, one reason for reduced performance of TopicLSA could be the fact that its term ranking and sentence ranking algorithms are not aligned, as discussed in section 3.1.

The comparisons we made above for each of the extraction rates show us that three of our proposed algorithms, JWTS, JWS and KIVSM-dot performed better than summarization systems proposed by other authors using topic modeling[14] for single-document summarization. JWTS performs the best at 30% rate, although JWS and KIVSM-dot achieve very close performance results with respect to JWTS. In the case of 20% and 25% extraction rates, JWS outperforms both JWTS and KIVSM-dot by some distance.

Of the proposed ranking algorithms, three of them (JWTS, JWS, and KIVSM-dot) favor long sentences for inclusion in summary while the rest (CWTS, CWS, KITS, and KIVSM-cos) normalize sentence length and give more or less equal chance to sentences of all lengths. Our experiments show that normalizing sentence length while generating scores significantly reduces the performance. This shows that longer sentences have more chance of capturing important points of the document. Unlike PROC3 of (Bhandari et al. 2008) which always selects the longest sentences, which still produces very competitive results, our algorithms (JWTS, JWS, and KIVSM-dot) do not always pick the longest sentences. This can be seen to be the case clearly for the algorithms of JWTS and JWS, as shown in sections 3.6.1.1 and 3.6.2.1 respectively. We have discussed how JWTS and JWS are related to PROC3. They weight the length of a sentence $p(s)$ by concept-based features specific to a sentence. This way, they are able to judge when a long sentence really does include useful concepts for inclusion in summaries. This has shown to be the case from our experiments, especially at lower extraction rates where PROC3 does not cope with the summarization task well.

[14] Note that the summarization system of first *n* sentences is the best performing system at the 20% extraction rate. It is not an approach based on topic modeling. It just serves as a strong baseline in the domain of news articles.

Chapter 5. Conclusion and Recommendations

5.1. Conclusion

This thesis investigated the application of topic modeling to the task of concept-based single-document Amharic text summarization. Text summarization involves the identification of topics of a document and the use of these topics to select sentences that best summarize the document. The identification of topics using the topic modeling approach results in word-based features and sentence-based features. From the extensive literature review we carried out, all but one of the previous works made use of the sentence-based features to form sentence scores. Only one work used word-based features for sentence ranking but the ranking algorithm it used is not quite sound from a theoretical point of view. The research highlighted the advantages of using the word-based features (keyword approach) over the sentence-based features (sentence-based approach).

We proposed new approaches to use the word-based features that result from a PLSA (Probabilistic Latent Semantic Analysis) model. PLSA was selected over other topic models, LSA (Latent Semantic Analysis) and LDA (Latent Dirichlet Allocation), because of its appropriateness for single-document summarization. In total, we proposed six algorithms, each algorithm categorized under one of three approaches:

- Approach 1
  - JWTS (joint word topic sentence) algorithm
  - CWTS (conditional word topic sentence) algorithm
- Approach 2
  - JWS (joint word sentence) algorithm
  - CWS (conditional word sentence) algorithm
- Approach 3
  - KITS (keywords in topic simplex) algorithm
  - KIVSM (keywords in vector space model) algorithm

In addition, to take advantage of the fact that we experiment with news articles, all of the algorithms always include the first sentence of the document while forming the document summaries. This is important because news articles put important points of the document near the beginning of the document in most cases. These six algorithms were used to explore the different possibilities of using the word-based features of a PLSA model for text summarization. Each algorithm consists of two major steps

- **Keyword selection**: words of the document that best represent the main topics/points of the document are selected.
- **Sentence scoring**: sentences of the document that best contain the keywords of the document are selected for inclusion in the summary.

The six algorithms differ by the approaches they use to carry out the two major steps:

- Keyword selection
  - Approach 1 selects keywords that are dominant within a single topic of the document only.
  - Approaches 2 and 3 select keywords that are dominant across all topics of the document.
- Sentence scoring
  - Approaches 1 and 2 have sentence scoring methods that estimate the probability of each sentence containing the keywords of the document.
  - Approach 3 first gives representation to the keywords like sentences of the document. Then it measures the degree of similarity between the resulting keywords representation and each sentence of the document.

We carried out experiments to determine which of the proposed algorithms works best for single-document Amharic text summarization. Experiments were also carried to compare the performance of the proposed algorithms with systems proposed by previous works. The algorithms were evaluated for precision/recall on three different extraction rates: 20%, 25% and 30%.

From the six algorithms we proposed, JWS performs the best. JWS selects keywords that are dominant across all topics and ranks these keywords based on a joint probability measure. JWS gives the best performance for 20% and 25% extraction rate summaries while JWTS gives the best performance for the 30% extraction rate summary.

When compared to previous summarization approaches using topic modeling, the best of the proposed systems gave improved results. This shows that the PLSA-based keyword approach to single-document text summarization represents a step forward in building better summarization systems using topic modeling.

5.2. Recommendations

We have explored the use of the keyword approach to Amharic text summarization in detail and obtained results that improve upon previous works. However, more investigation should be conducted to advance the technology of Amharic text summarization further. We suggest potential research directions below:

- Applying the proposed algorithms for the task of multi-document, query-focused and update summarization tasks is one possible future work. These summarization tasks usually involve the summarization of a collection of topically related documents. This means that redundant materials are likely to exist throughout the document collection whose summary is required. Thus, unlike the case of single-document summarization, the former summarization tasks require a component that detects and removes redundant material from the summary.
- The stemming algorithm used in our work identifies inflectional morphology only. However, we believe that the increase in co-occurrence counts that results by the use of derivation morphology could result in increased performance. Especially for the task of single document summarization where co-occurrence counts are scarce, the proposed summarization algorithms could benefit from more co-occurrence data to work with.
- It is quite common that LSA based systems make use of term weighting to help the summarization algorithms perform better. However, all of the summarization systems we reviewed that use probabilistic topic models, including ours, do not use term weighting.

Thus, future works should investigate the appropriate term weighting techniques for probabilistic topic models.

- Local languages like Amharic suffer from lack of linguistic resources that can help in representing the various concepts of documents. Thus, it becomes necessary to maximize the use of the co-occurrence statistics available for documents. One way of achieving this is to consider the use of larger units of co-occurrence counts such as bigrams, trigrams; their "skip" versions (e.g. skip bi-grams); and grams that are considered the same if the order of the words they contain is ignored. These statistics can be used together with the simple unigram counts we considered. The use of such co-occurrence statistics although possible for single-document summarization, we think that it is more advantageous for multi-document summarization. This is because of the possibility for the appearance of high-order grams in multiple documents to be greater than that for single documents.
- The algorithms proposed here generated sentence/keywords features and use these features for creating sentence scores. If we take the case of the algorithms JWTS and CWTS, the effect of the keywords on resulting sentence scores degenerates because of their scoring technique (multiplication of sentence features). This in turn resulted in reduced performance of the algorithms when compared to JWS and CWS. To solve this problem, we can resort to other ways of combining the generated features for creating scores. For example, we can use addition to combine the features when creating the sentence scores.
- The establishment of a large-scale standard data set for evaluation of Amharic text summarization systems is crucial. This helps to greatly simplify comparing works that have been done before with future works. The construction of the data set for multi-document summarization, query-focused summarization and update summarization should particularly be given attention. The presence of such a data set would encourage more research to be carried out on the mentioned types of summarization, as current works are dominated by single-document text summarization.

References

Abraham Adefris. 2007. “Automatic Multi-Source Amharic News Summarization.”, MSc thesis, Graduate School of Telecommunications & Information Technology, Addis Ababa, Ethiopia.

Arora, R. and B. Ravindran. 2008. “Latent dirichlet allocation based multi-document summarization.” In *Proc. of AND ’08*: 91–97.

Barzilay, Regina and Michael Elhadad. 1997. “Using lexical chains for text summarization.” In *Proceedings of the ACL/EACL’97 Workshop on Intelligent Scalable Text Summarization*: 10–17, Madrid, Spain.

Bergler, Sabine, René Witte, Michelle Khalife, Zhuoyan Li, and Frank Rudzicz. 2003. “Using knowledge-poor coreference resolution for text summarization.” In *DUC Workshop on Text Summarization,* May-June.

Bhandari, H., M. Shimbo, T. Ito, and Y. Matsumoto. 2008. “Generic text summarization using probabilistic latent semantic indexing.” In *Proc. of IJCNLP*.

Blei, D. M., A. Y. Ng, M. I. Jordan, and J. Lafferty (editor). 2003. “Latent Dirichlet Allocation.”, In *Journal of Machine Learning Research*, 3: 993-1022.

Blei, David M. and John D. Lafferty. 2005. “Correlated topic models.” In *NIPS*.

Boguraev, B., and C. Kennedy. 1997. “Salience-based content characterization of text documents.” In *ACL/EACL-97 Workshop on Intelligent Scalable Text Summarization*: 2-9, Madrid, Spain.

Bosma, Wauter. 2008. “Discourse oriented summarization.” PhD diss., Center for Telematics and Information Technology, Netherlands.

Brants, T. 2002. “Test data likelihood for PLSA models.” In *ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, Tampere, Finland.

Brants, T., F. Chen, and I. Tsochantaridis. 2002. “Topic-based document segmentation with probabilistic latent semantic analysis.” In *Proc. of CIKM ’02* :211–218.

Brin, S. and L. Page. 1998. “The anatomy of a large-scale hypertextual Web search engine.” In *Computer Networks and ISDN Systems*, 30: 1–7.

Carbonell, J., and J. Goldstein. 1998. “The use of MMR, diversity-based reranking for reordering documents and producing summaries.” In *Proc. ACM SIGIR.*

Cohn, D., H. Chang. 2001. “Learning to probabilistically identify authoritative documents.” In *Proceedings of 18th International Conference of Machine Learning*.

Daniel Hailu. 2006. “Automatic Amharic Text Summarization.” MSc thesis, Graduate School of Telecommunications & Information Technology, Addis Ababa, Ethiopia, 2006.

Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. “Indexing by Latent Semantic Analysis.” In *Journal of the American Society for Information Science*, 41 (6): 391-407.

Ding, Chris H. Q. 2005. “A Probabilistic Model for Latent Semantic Indexing.” In *Journal of the American Society for Information Science and Technology*, 56 (6): 597–608.

Donaway, R., K. Drummey, and L. Mather. 2000. “A comparison of rankings produced by summarization evaluation measures.” In *NAACL-ANLP Workshop on Automatic Summarization*.

Dong, Z., and Q. Dong. 2003. "HowNet - a hybrid language and knowledge resource." In *The proceedings of International Conference on Natural Language Processing and Knowledge Engineering*: 820 – 824.

Edmundson, H.P. 1969. “New Methods in Automatic Extracting.” In *Journal of the Association for Computing Machinery*, 16 (2): 264-285.

Erkan, G., and D. R. Radev. 2004. “Lexrank: Graph-based Centrality as Salience in Text Summarization.” In *Journal of Artificial Intelligence Research (JAIR)*.

Filatova, E. and V. Hatzivassiloglou. 2003. “Domain-independent detection, extraction, and labeling of atomic events.” In *Proceedings of RANLP*: 145-152, Borovetz, Bulgaria.

Filatova, E. and V. Hatzivassiloglou. 2004. “Event-based extractive summarization.” In *ACL-04*: 104-111.

Girolami, Mark and Ata Kaban. 2003. “On an Equivalence between PLSI and LDA.” In *Proceedings of ACM SIGIR’03*, Toronto, Canada.

Gong, Y., and X. Liu. 2001. “Generic text summarization using relevance measure and latent semantic analysis.” In *Proc. of SIGIR ’01*: 19–25.

Griffiths, T. L., and M. Steyvers. 2004. “Finding scientific topics.” In *Proceedings of the National Academy of Science*, 101:5228-5235.

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. “The WEKA data mining software: An Update.” In *SIGKDD Explorations*, 11(1).

Harabagiu, S. M., and F. Lacatusu. 2002. “Generating single and multi-document summaries with GISTEXTER.” In *Document Understanding Conference*.

Helen Adane. 2006. “Text Summarization on Amharic Legal Judgments.” MSc thesis, Addis Ababa University, Addis Ababa, Ethiopia.

Hennig, Leonhard. 2009. “Topic-based multi-document summarization with probabilistic latent semantic analysis.” In *Recent Advances in Natural Language Processing,RANLP*.

Hofmann, T. 1999a. “Probabilistic latent semantic analysis.” In *Uncertainty in Artificial Intelligence, UAI ’99*: 50–57.

Hofmann, T. 1999b. “Probabilistic latent semantic indexing.” In *Proc. of SIGIR ’99*: 50–57.

Hofmann, T. 2001. “Unsupervised Learning by Probabilistic Latent Semantic Analysis.” In *Machine Learning*, 42: 177–196, 2001

Hofmann, T., and Jan Puzicha. 1998. “Unsupervised learning from dyadic data.” *Technical Report TR-98-042*, International Computer Science Institute, Berkeley, California.

Hovy, Eduard, and Chin-Yew Lin. 1999. “Automated text summarization in SUMMARIST.” In *Advances in Automatic Text Summarization*: 81–94, MIT Press, Cambridge MA, USA.

Hovy, Eduard. 2005. “Text summarization”, In *The Oxford Handbook of Computational Linguistics* : 583-598.

Jones, Karen Spärck . 2007. “Automatic summarizing: a review and discussion of the state of the art.” *Technical Report UCAM-CL-TR-679*, University of Cambridge.

Kamil Nuru. 2004. “Automatic Amharic News Text Summarization.”, MSc thesis, Addis Ababa University, Addis Ababa, Ethiopia.

Kleinberg, J.M. 1999. “Authoritative Sources in a Hyperlinked Environment.” In *Journal of the ACM*, 46(5): 604–632.

Ledeneva, M. en C. Yulia Nikolaevna. 2008. “Automatic Language-Independent Detection of Multiword Descriptions for Text Summarization.” PhD diss., National Polytechnic Institute Mexico.

Li, W. and A. McCallum. 2006. “Pachinko allocation: Dag-structured mixture models of topic correlations.” In *Proceedings of the 23rd international conference on Machine learning*, 23:577-584.

Lin, C. Y. 2004. “Looking for a Few Good Metrics: Automatic Summarization Evaluation – How Many Samples Are Enough?” In *Proc. of NTCIR Workshop*, Japan.

Luhn, H.P. 1958. “The Automatic Creation of Literature Abstracts.”, In *IBM Journal of Research and Development*, 2 (2): 159-165.

Mani, I. 2001. *Automatic Summarization*, *Volume 3 of Natural language processing.*, Amsterdam, the Netherlands: John Benjamins.

Mann, W. C. and S. A. Thompson. 1988. “Rhetorical Structure Theory: Toward a Functional Theory of Text Organization”, In *Text - Interdisciplinary Journal for the Study of Discourse*, 8 (3): 243-281.

Marcu, Daniel. 1997a. “The rhetorical parsing, summarization, and generation of natural language texts.” PhD diss., Department of Computer Science, University of Toronto, Forthcoming.

Marcu, Daniel. 1997b. “From discourse structures to text summaries”. In *Proceedings of the ACL’97/EACL’97 Workshop on Intelligent Scalable Text Summarization*: 82–88, Madrid, Spain.

Melese Tamiru. 2009. “Automatic Amharic Text Summarization Using Latent Semantic Analysis.” MSc thesis, Addis Ababa University, Addis Ababa, Ethiopia.

Mihalcea, R. and P. Tarau. 2004a. “TextRank – bringing order into texts.” In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spain.

Mihalcea, R., and P. Tarau. 2004b. “A language independent algorithm for single and multiple document summarization.” In *Proceedings of Second International Joint Conference on Natural Language Processing*: 19-24.

Miller, G. A. 1995. “WordNet: a lexical database for English.” In *Communications of the ACM*, 38 (11): 39–41.

Nega Alemayehu and Peter Willett. 2002. “Stemming of Amharic Words for Information Retrieval”, In *Literary and Linguistic* Computing, 17 (1).

Nenkova, A. 2005.” Automatic text summarization of newswire: Lessons learned from the document understanding conference.” In *Proceedings of AAAI* 2005, Pittsburgh, USA.

Nenkova, Ani , Lucy Vanderwende, and Kathleen McKeown. 2006. “A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization.” In *Proceedings of ACM SIGIR*, Seattle, Washington, USA.

Nenkova, Ani. 2006. “Summarization evaluation for text and speech: issues and approaches.” In *ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh*, PA, USA.

Ono, Kenji, Kazuo Sumita and Seiji Miike. 1994. “Abstract generation based on rhetorical structure extraction.” In *Proceedings of the 15th International Conference on Computational Linguistics*, 1: 344–384, Kyoto, Japan, 1994.

Otterbacher, Jahna, Gunes Erkan, and Dragomir R. Radev. 2005.“Using random walks for question-focused sentence retrieval”, In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing* pp: 915 – 922.

Radev, D. R., E. H. Hovy, and K. McKeown. 2002. “Introduction to the special issue on summarization.” *Computational Linguistics*, 28 (4): 399–408.

Radev, Dragomir R., Simone Teufel, Horacio Saggion, Wai Lam, John Blitzer, Hong Qi, Arda Celebi, Danyu Liu, and Elliott Drabek. 2003. “Evaluation challenges in large-scale document summarization.” In *Proceeding of the 41st meeting of the Association for Computational Linguistics*: 375–382, Sapporo, Japan.

Richardson, M., P. Domingos. 2002. “The intelligent surfer: probabilistic combination of link and content information in PageRank.” In *Advances in Neural Information Processing Systems.*

Roelleke, T. 2003. “A frequency-based and a poisson-based definition of the probability of being informative.” In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval.*

Salton, G., A. Wong, and C. S. Yang. 1975. “A vector space model for automatic indexing.” In *Communication of the ACM*, 18 (11): 613-620.

Silber, H.G. and K.F. McCoy. 2002. “Efficiently computed lexical chains as an intermediate representation for automatic text summarization”, In *Computational Linguistics*, 28 (4): 487-496.

Steinberger, J., and K. Ježek. 2004. “Using latent semantic analysis in text summarization and summary evaluation.” In *Proc. ISIM ’04*: 93–100.

Steinberger, J., M. Poesio, M. kabadjov, and K. Ježek. 2007. “Two Uses of Anaphora Resolution in Summarization.” In *Information Processing and Management: an International Journal*, 43 (6).

Steinberger, Josef. 2007. “Text Summarization within the LSA Framework.” PhD diss. University of West Bohemia.

Steyvers, M., and T. Griffiths. 2007. “Probabilistic Topic Models.” In *Latent Semantic Analysis: A Road to Meaning*, edited by T. Landauer, D McNamara, S. Dennis, and W. Kintsch. Laurence Erlbaum.

Teferi Andargie. 2005. “The Application of Machine Learning Technique (NAÏVE BAYES) for Automatic Text summarization (The Case of Amharic News Texts).”, MSc thesis, Addis Ababa University, Addis Ababa, Ethiopia.

Tessema Mindaye Mengistu. 2007. “Design and Implementation of Amharic Search Engine.” Msc thesis, Addis Ababa University, Addis Ababa.

Winta Aklok. 2009. “Multi-Source Amharic News Summarization Using Machine Learning Approach (Artificial Neural Network).”, MSc thesis, Graduate School of Telecommunications & Information Technology, Addis Ababa, Ethiopia.

Yeh, Jen-Yuan, Hao-Ren Ke, Wei-Pang Yang, and I-Heng Meng. 2005. “Text summarization using a trainable summarizer and latent semantic analysis.” In *Special issue of Information Processing and Management on An Asian digital libraries perspective*, 41 (1): 75–95.

Appendix A. Ideal Summary Preparation

This appendix describes guideline and instructions that are used to prepare ideal summaries.

A.1 Guideline for Ideal Summary Preparation

You are requested to form an extractive summary for each of the documents you are given. An extractive summary is created by selecting a certain number of sentences that are judged to be the most important out of the original text.

When sentences are selected for inclusion in the summary all that needs to be considered is the importance of the sentences. When the selected sentences are concatenated together to form the summary they **do not** need to be coherent. Among other things, this means that you can select a sentence for inclusion in a summary and leave out a sentence which is not that important but contains information that describes another selected sentence. Thus this work **does not** deal with creating coherent summaries.

Please follow these two methods of selecting sentences for inclusion in the summary.

1. Delete material that is trivial (not important). Example: unimportant details.
2. Delete material that, although it is important, is also redundant.

A.2 Instructions for Ideal Summary Preparation

Each sentence in a document is identified by a serial number. To make the summarization process easier, each sentence starts on a new line. Select 30%, 25% and 20% of the total number of the sentences in the document. These numbers are already calculated for each document and put in a table at the end of the document. Each column of the table represents spaces where you should write the serial number of sentences that are selected for a particular extraction rate (30%, 25%, or 20%). The number of sentences to be selected is indicated at the top of each column. Starting from the left of the table, the second column represents sentences at 30% extraction rate, the third one at 25% extraction rate and the last one at 20%. Note that within a column of chosen sentences, ranking is not important. The serial numbers shown in the leftmost column of the

table are just to show you how many sentences you have selected, not the importance of the sentences.

A recommended method to generate sentences at 30% extraction rate is as follows

- As you read the sentences for the first time or as you are rereading them, categorize the sentences by marking them (see the third bullet to see a recommended method for marking the sentences) in one of three groups: selected, candidate, discarded. This would be advantageous for keeping memory of the status of each sentence for inclusion in the summary in a reliable manner.
- Make sure to change the label of a sentence when its category is changed to guarantee accurate summarization.
- To mark the sentences, the following methods are recommended
    - If you are using an electronic version of the document: use boldface for selected sentences, underline candidate sentences and mark discarded sentences with red color.
    - If you are using printed version of the document: mark the serial number of a sentence by a tick sign (✓) if it is selected, circle the serial number if it is a candidate, or with a cross sign (×) if it is discarded.

To generate summary sentences at each of the three extraction rates, the following method is recommended

1. First, choose sentences at 30% extraction rate according to the guideline of section A.1 (giving attention to the two methods indicated).
2. Second, eliminate the required number of sentences from the summary produced in step one using the guideline of section A.1 (giving attention to the two methods indicated) to reach at 25% extraction rate summary sentences.
3. Finally, do the same as in step two to reach at the 20% rate summary sentences.

Appendix B. List of Short Words and their Expanded Forms

ት/ቤት	ትምህርት ቤት
ት/ርት	ትምህርት
ት/ክፍል	ትምህርት ክፍል
ሃ/አለቃ	ሀምሳ አለቃ
ሃ/ስላሴ	ሀይለ ስላሴ
ደ/ዘይት	ደብረ ዘይት
ደ/ታቦር	ደብረ ታቦር
መ/ር	መምህር
መ/ቤት	መስሪያ ቤት
መ/አለቃ	መቶ አለቃ
ክ/ከተማ	ክፍለ ከተማ
ክ/ሀገር	ክፍለ ሀገር
ወ/ር	ወታደር
ወ/ሮ	ወይዘሮ
ወ/ሪት	ወይዘሪት
ወ/ስላሴ	ወሌደ ስላሴ
ፍ/ስላሴ	ፍቅረ ስላሴ
ፍ/ቤት	ፍርድ ቤት
ጽ/ቤት	ጽህፈት ቤት
ሲ/ር	ሲስተር
ፕ/ር	ፕሮፌሰር
ጠ/ሚኒስትር	ጠቅላይ ሚኒስትር
ዶ/ር	ዶክተር
ገ/ገዮርጊስ	ገብረ ገዮርጊስ
ቤ/ክርስትያን	ቤተ ክርስትያን
ም/ስራ	ምክትል ስራ
ም/ቤት	ምክር ቤት
ተ/ሃይማኖት	ተክለ ሃይማኖት
ሚ/ር	ሚኒስትር
ኮ/ሌ	ኮሎኔሌ
ሜ/ጀነራል	ሜጀር ጀነራል
ብ/ጀነራል	ብርጋዴር ጀነራል
ሌ/ኮለኔሌ	ሌተናል ኮለኔል
ሊ/መንበር	ሊቀ መንበር
አ/አ	አዲስ አበባ
ር/መምህር	ርእሰ መምህር
ፕ/ት	ፕሬዝዳንት
ዓ.ም	አመተ ምህረት
ዓ.ዓ	አዲስ አበባ
ዶ.ር	ዶክተር

Appendix C. List of Stop-Words

ሁሉ
ሁሉም
ኋላ
ሁኔታ
ሆነ
ሆኑ
ሆኖም
ሁል
ሁሉንም
ላይ
ሌላ
ሌሎች
ልዩ
መሆኑ
ማለት
ማለቱ
መካከል
የሚገኙ
የሚገኝ
ማድረግ
ማን
ማንም
ሰሞኑን
ሲሆን
ሲል
ሲሉ
ስለ
ቢቢሲ
ቢሆን
ብለዋል
ብቻ
ብዛት
ብዙ
ቦታ
በርካታ
በሰሞኑ
በታች
በኋላ
በኩል
በውስጥ
በጣም
ብቻ
በተለይ
በተመለከተ
በተመሳሳይ
የተለያየ
የተለያዩ
ተባለ
ተገለጸ
ተገልጿል
ተጨማሪ
ተከናውኗል
ችግር
ታች
ትናንት
ነበረች
ነበሩ
ነበረ
ነው
ነይ
ነገር
ነገሮች
ናት
ናቸው
አሁን
አለ
አስታወቀ
አስታውቀዋል
አስታውሰዋል
እስካሁን
አሳሰበ
አሳስበዋል
አስፈሊጊ
አስገነዘቡ
አስገንዝበዋል
አብራርተዋል
አበራርተው
አስረድተዋል
እስከ
እባክህ
እባክሽ
እባክዎ
አንድ
አንጻር
እስኪደርስ
እንኳ
እስከ
እዚሁ
እና
እንደ
እንደገለጹት
እንደተገለጸው
እንደተናገሩት
እንደአስረዱት
እንደገና
ወቅት
እንዲሁም
እንጂ
እዚህ
እዚያ
እያንዳንዱ
እያንዳንዳችው
እያንዳንዷ
ከ
ከኋላ
ከላይ
ከመካከል
ከሰሞኑ
ከታች
ከውስጥ
ከጋራ
ከፊት
ወዘተ
ወይም
ወደ
ዋና
ወደፊት
ውስጥ
ውጪ
ያለ
ያሉ
ይገባል
የኋላ
የሰሞኑ
የታች
የውስጥ
የጋራ
ያ
ይታወሳል
ይህ
ደግሞ
ደረስ
ጋራ
ግን
ገሌጿል
ገልጸዋል
ግዜ
ጥቂት
ፊት
ደግሞ
ዛሬ
ጋር
ተናግረዋል
የገለጹት
ይገልጻል
ሲሉ
ብለዋል
ስለሆነ
አኖ
ሆኖም
መግለጹን
አመልክተዋል
ይናገራለ

Appendix D. List of Suffixes and Prefixes

Suffixes

ቱ	ችው	ባቸው
ዎች	ው	ባችሁ
ም	ሁ	ቱ
ች	ኝ	ሽ
ን	ና	ይቱ
ዬ	ለት	ዎቹ
ዎ	ላት	የው
ህ	ላቸው	ኞች
ሽ	ላችሁ	ያል
ዋ	በት	ኛ
ችን	ባት	ቸው

Prefixes

እየ	ለ	እስኪ
ሲ	ይ	እንድ
የ	እንዲ	ከነ
የሚ	ስለ	እን
ከ	እስከ	እነ
በ	እንደ	

Appendix E. Java Class for JWTS

```
import java.io.*;
import Jama.Matrix;
import java.util.TreeMap;

public class JWTS {

public double logL; //Stores predictive log-likelihood of the current PLSA
                    //model for use by the "score" method

public double []  rank (Matrix termBySent, int numTopic, double cc, int
                        numRestart, double beta) throws Exception
{

//cc stands for convergence criteria (%)

plsa plsa1 = new plsa (numTopic,cc,numRestart,beta);
TreeMap <String, Matrix> para = plsa1.trainp(termBySent.copy());

double [] ppdz = new double [termBySent.getColumnDimension()];

double product;

for(int i=0;i<para.get(plsa.sentByTopicf).getRowDimension();i++)
{
      product = 1.0d;

      for(int j=0;j < para.get(plsa.sentByTopicf).getColumnDimension();j++)
      {
      product = product * (para.get(plsa.sentByTopicf).get(i, j));
      }

     ppdz[i]=product;

}
logL = plsa1.logL2(para, termBySent.copy());
return ppdz;

}  //End of class "rank"


public int [] score (String position, int numModel, double rate,File doc,
                     int numTopic, double cc, int numRestart, double beta)
                     throws Exception
{
algoUtil util = new algoUtil();
double [] score;
double [] tempScore;
double [][] tempScores = new double [numRestart][];
double [] logLs = new double [numRestart];
int [] ixLogLs = new int [numRestart];  // "ix" stands for index

Matrix termBySent;
sentencevector sv= new sentencevector();
```

```
termBySent= sv.generatematrix(doc);

//Calcualte number of sentences extracted
int len = (int) Math.round(((double) termBySent.getColumnDimension())*rate);
///////Finished ... //////////

score = new double [termBySent.getColumnDimension()];
util.setToZeros(score);

//numRestart number of restarts
if(position.equals(algoUtil.all))
{
      System.out.println(numRestart + " restarts.");
      for (int i=0; i<numRestart;i++)
      {
            tempScore = rank(termBySent.copy(), numTopic,cc,1,beta);
            score = util.plus(score, tempScore);
      }
}

//Top models
else if(position.equals(algoUtil.top))
{
      System.out.println("Top "+numModel+" of "+numRestart+" restarts.");

      for (int i=0; i<numRestart;i++)
      {
            tempScores[i] =  rank(termBySent.copy(), numTopic,cc,1,beta);
            logLs[i] = logL;
      }
      ixLogLs = util.bubbleSort(logLs) ;

      for (int i=0; i<numModel ;i++)
      {
            score = util.plus(score, tempScores[ixLogLs[i]]);

      }

}

//Bottom models
else if(position.equals(algoUtil.bottom))
{

      System.out.println("Top "+numModel+" of "+numRestart+" restarts.");
      for (int i=0; i<numRestart;i++)
      {
            tempScores[i] =  rank(termBySent.copy(), numTopic,cc,1,beta);
            logLs[i] = logL;
      }
      ixLogLs = util.bubbleSort(logLs) ;
      util.reverse(ixLogLs);
      for (int i=0; i<numModel ;i++)
      {
            score = util.plus(score, tempScores[ixLogLs[i]]);
      }
```

```
}


else
{
      //Dummy code. No scoing method is executed here.
      //Gives “First n Sentences” algorithm
      System.out.println("Dummy scoring method");
      score = new double [termBySent.getColumnDimension()];
}

System.out.println("Sentence socres");
util.display(score);

int [] tempSummary = util.bubbleSort(score);
int [] summary = new int [len];
for (int i=0; i<len; i++)
{
      summary [i] = tempSummary [i] + 1;//Because java index starts from zero
                                        //while sentence number starts from one
}

util.addOne(summary); //Include the first sentence if it is not there
System.out.println("Ranked sentences of " + len + " in total at " +
                  (int)(rate*100)+"% extraction rate.");
for (int i=0; i< summary.length;i++)
{
      System.out.print(summary[i] + ", ");
}
System.out.println();

return summary

} //End of class “score”


public void expJWTS (String [] file, String outputFileName,String
                     position,int numModel,double rate, int numTopic, double
                     cc, int numRestart, double beta) throws Exception
{
String dirPr=algoUtil.prDir.concat("\\expJWTS\\"+Double.toString(rate)+"\\");
String dirPr25 = algoUtil.prDir.concat("\\expJWTS\\0.25\\");
String dirPr20 = algoUtil.prDir.concat("\\expJWTS\\0.2\\");

algoUtil util = new algoUtil();
String fileName;


double [][] pr = new double [file.length][3];//pr stands for precision/recall
double [][] pr25 = new double [file.length][3];
double [][] pr20 = new double [file.length][3];
int [] summary;
double [] avgPr;
double [] avgPr25 = new double[0]; //Dummy initialization to obey compiler
double [] avgPr20 = new double [0];
```

```
//Summarize at given rate (usaully 0.3d)
for (int i=0; i<file.length;i++)
{
      fileName = algoUtil.fileDir.concat(file[i]+".txt");
      summary = score(position,numModel,rate,new
                  File(fileName),numTopic,cc,numRestart,beta);
      pr[i]=util.pr(summary.clone(), file[i], rate);
      if(rate == algoUtil.rate30)
      {
      pr25[i]=util.pr(summary.clone(), file[i], algoUtil.rate25);
      pr20[i]=util.pr(summary.clone(), file[i], algoUtil.rate20);
      }

}

//Calculate average p/r
avgPr = util.avg(pr.clone());
if(rate == algoUtil.rate30)
{
      avgPr25 = util.avg(pr25.clone());
      avgPr20 = util.avg(pr20.clone());

}

//Writing avgPr to file
String prFile;
String signature = "position="+ position+" numModel="+numModel+"
                  rate="+rate+" numTopic="+numTopic+" cc="+cc+"
                  numRestart="+numRestart+" beta="+beta;
prFile = dirPr.concat(outputFileName+".txt");

util.writeAvg(prFile, signature, avgPr.clone());


if(rate == algoUtil.rate30)
{

      signature = "position="+ position+" numModel="+numModel+"
                  rate="+algoUtil.rate25+" numTopic="+numTopic+" cc="+cc+"
                  numRestart="+numRestart+" beta="+beta;
      prFile = dirPr25.concat(outputFileName+".txt");
      util.writeAvg(prFile, signature, avgPr25.clone());

      signature = "position="+ position+" numModel="+numModel+"
                  rate="+algoUtil.rate20+" numTopic="+numTopic+" cc="+cc+"
                  numRestart="+numRestart+" beta="+beta;
      prFile = dirPr20.concat(outputFileName+".txt");
      util.writeAvg(prFile, signature, avgPr20.clone());


}
} //End of class “expJWTS”

} //End of class “JWTS”
```

Appendix F. Full Experimental Results for Selected Experiments

Table F.1: Results of experiment 1.1. It contains tables F.1.1, F.1.2 and F.1.3.

Table F.1.1: Result of experiment 1.1 for

$\beta = 0.8, Number\ of\ topics = 2\ and\ convergence\ criterion\ (\%) = 0.03.$

No.	Scoring Method	Precision/Recall		
		Extraction Rate (%)		
		30	25	20
1	1 random start	0.42531	0.40276	0.38904
2	Top 1 of 20 restarts	0.42055	0.39402	0.37072
3	Bottom 1 of 20 restarts	0.44601	0.41885	0.39891
4	5 restarts	0.45563	0.42267	0.41102
5	Top 5 of 20 restarts	0.46097	0.42837	0.40388
6	Bottom 5 of 20 restarts	0.46540	0.43830	0.41156
7	10 restarts	0.47691	0.43805	0.41532
8	Top 10 of 20 restarts	0.46151	0.42795	0.40540
9	Bottom 10 of 20 restarts	**0.48668**	0.45203	**0.42707**
10	15 restarts	0.48499	0.45185	0.42337
11	Top 15 of 20 restarts	0.47857	0.43408	0.41322
12	Bottom 15 of 20 restarts	0.47783	0.44904	0.41851
13	20 restarts	0.48538	**0.45228**	0.41672

Table F.1.2: Result of experiment 1.1 for

$\beta = 0.7, Number\ of\ topics = 4\ and\ convergence\ criterion\ (\%) = 0.9.$

No.	Scoring Method	Precision/Recall Extraction Rate (%)		
		30	25	20
1	1 random start	0.47159	0.43943	0.40959
2	Top 1 of 20 restarts	0.46620	0.42735	0.40049
3	Bottom 1 of 20 restarts	0.48171	0.44819	0.40586
4	5 restarts	0.49614	0.45978	0.42637
5	Top 5 of 20 restarts	0.48863	0.43973	0.40115
6	Bottom 5 of 20 restarts	0.49412	0.46298	0.42679
7	10 restarts	0.49756	0.45600	0.42329
8	Top 10 of 20 restarts	0.49043	0.45242	0.41955
9	Bottom 10 of 20 restarts	0.49106	0.45732	0.42000
10	15 restarts	0.50583	**0.46670**	0.43010
11	Top 15 of 20 restarts	0.49691	0.46052	**0.43149**
12	Bottom 15 of 20 restarts	**0.50586**	0.46299	0.42175
13	20 restarts	0.49864	0.45872	0.41709

Table F.1.3: Result of experiment 1.1 for

$\beta = 0.9, Number\ of\ topics = 6\ and\ convergence\ criterion\ (\%) = 0.007.$

No.	Scoring Method	Precision/Recall Extraction Rate (%)		
		30	25	20
1	1 random start	0.43457	0.39537	0.36958
2	Top 1 of 20 restarts	0.44245	0.40555	0.38100
3	Bottom 1 of 20 restarts	**0.45086**	**0.42643**	**0.39982**
4	5 restarts	0.43625	0.40176	0.38542
5	Top 5 of 20 restarts	0.42332	0.38988	0.36843
6	Bottom 5 of 20 restarts	0.43532	0.40531	0.39207
7	10 restarts	0.43006	0.40170	0.37166
8	Top 10 of 20 restarts	0.42059	0.39338	0.36823
9	Bottom 10 of 20 restarts	0.43361	0.40682	0.38651
10	15 restarts	0.43028	0.40908	0.38661
11	Top 15 of 20 restarts	0.42587	0.40034	0.38006
12	Bottom 15 of 20 restarts	0.43513	0.40317	0.38392
13	20 restarts	0.42895	0.40162	0.38437

Table F.2: Results of experiment 1.2. It contains tables F.2.1, F.2.2 and F.2.3.

Table F.2.1: Result of experiment 1.2 when the scoring method is 15 restarts.

No.	β	Number of Topics	Precision/Recall								
			Convergence Criterion (%)								
			0.9			0.03			0.007		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20
1	1	2	0.50768	0.46336	0.43160	0.45456	0.43006	0.41545	0.44418	0.42829	0.41588
2		3	0.47579	0.44152	0.42180	0.42573	0.40132	0.37740	0.41354	0.38846	0.36701
3		4	0.45332	0.42688	0.38383	0.43025	0.39920	0.38522	0.41597	0.40374	0.37418
4		5	0.46301	0.43601	0.42589	0.43757	0.40603	0.37372	0.41740	0.38864	0.37611
5		6	0.44444	0.41650	0.38720	0.42719	0.39887	0.38241	0.42025	0.39909	0.36059
6	0.95	2	0.49904	0.44991	0.43360	0.45254	0.41923	0.39815	0.43782	0.40728	0.37058
7		3	0.49052	0.44486	0.42316	0.42366	0.38520	0.36777	0.41886	0.39721	0.39065
8		4	0.46847	0.43555	0.41027	0.44019	0.41642	0.39214	0.40984	0.37999	0.34733
9		5	0.46542	0.44558	0.41088	0.43532	0.41236	0.38055	0.42270	0.39099	0.36206
10		6	0.46445	0.41304	0.39515	0.43112	0.41039	0.38642	0.42845	0.39124	0.37977
11	0.9	2	0.50272	0.46924	**0.43530**	0.46285	0.42800	0.39922	0.45611	0.41782	0.40870
12		3	0.49205	0.45455	0.42392	0.41988	0.39551	0.37044	0.43934	0.39735	0.36769
13		4	0.47555	0.44753	0.40809	0.42150	0.40605	0.38154	0.42239	0.38233	0.37812
14		5	0.46055	0.41480	0.40009	0.43158	0.40152	0.37515	0.42923	0.40525	0.38649
15		6	0.46707	0.42405	0.41028	0.44063	0.40705	0.37550	0.43225	0.40254	0.37758
16	0.85	2	0.51124	0.46207	0.43034	0.47324	0.43924	0.41553	0.46093	0.43290	0.41461
17		3	0.49645	**0.47098**	0.42509	0.45300	0.42533	0.40033	0.44035	0.40914	0.39155
18		4	0.48847	0.44763	0.41023	0.44515	0.42229	0.38274	0.42528	0.40903	0.37017
19		5	0.47374	0.43643	0.40716	0.44060	0.39801	0.39506	0.45013	0.42300	0.40103
20		6	0.48193	0.43953	0.42763	0.44608	0.40194	0.36263	0.43345	0.39891	0.38336
21	0.8	2	0.50120	0.45411	0.42491	0.48210	0.45086	0.42591	0.47814	0.43249	0.41130

No.	β	Number of Topics	Precision/Recall								
			Convergence Criterion (%)								
			0.9			0.03			0.007		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20
22		3	0.49655	0.44990	0.42434	0.46521	0.42041	0.38587	0.45502	0.42748	0.39558
23		4	0.48076	0.43743	0.41987	0.44989	0.42036	0.38185	0.45496	0.42655	0.39068
24		5	0.47136	0.42957	0.40105	0.46405	0.42250	0.39617	0.44032	0.40926	0.38445
25		6	0.46416	0.43395	0.40982	0.44694	0.42026	0.39494	0.45158	0.42082	0.39844
26	0.75	2	0.50645	0.45707	0.42338	0.48421	0.45128	0.42759	0.48388	0.44873	0.41382
27		3	0.49529	0.45662	0.42183	0.46616	0.43533	0.40606	0.46828	0.43300	0.39737
28		4	0.49000	0.44682	0.42592	0.45968	0.42202	0.39834	0.46045	0.43790	0.40525
29		5	0.48111	0.44924	0.41322	0.46163	0.41928	0.38899	0.47111	0.42181	0.39074
30		6	0.47872	0.44296	0.40730	0.45986	0.42238	0.39485	0.45613	0.42745	0.39317
31	0.7	2	0.50445	0.46491	0.42463	0.48962	0.45885	0.43286	0.49660	0.44349	0.42648
32		3	0.50889	0.46640	0.42249	0.48079	0.44664	0.42191	0.48669	0.44568	0.42164
33		4	0.49511	0.45679	0.42068	0.47948	0.44228	0.41360	0.48563	0.45215	0.41341
34		5	0.47908	0.45247	0.42075	0.46545	0.43708	0.40043	0.47501	0.42885	0.40833
35		6	0.48024	0.44425	0.40792	0.46794	0.43774	0.42168	0.46760	0.43863	0.39898
36	0.65	2	0.51203	0.45537	0.41926	0.48845	0.45789	0.42789	0.48777	0.45773	0.41797
37		3	**0.51335**	0.45838	0.42968	0.47463	0.43639	0.41474	0.48521	0.44915	0.42476
38		4	0.50995	0.45876	0.42561	0.47511	0.45062	0.41985	0.48476	0.43290	0.41537
39		5	0.50719	0.46119	0.42608	0.46768	0.43571	0.40704	0.48633	0.44240	0.41670
40		6	0.50140	0.46319	0.42141	0.48754	0.45701	0.41927	0.48384	0.44242	0.41475
41	0.6	2	0.50829	0.45407	0.42478	0.49396	0.45677	0.43068	0.49617	0.45545	0.42279
42		3	0.50814	0.45438	0.43008	0.48508	0.45516	0.42681	0.49207	0.44513	0.42723
43		4	0.50928	0.45979	0.43414	0.48751	0.45305	0.42577	0.48914	0.44491	0.40636
44		5	0.50698	0.46075	0.42465	0.49076	0.45268	0.43017	0.48329	0.44856	0.41248

No.	β	Number of Topics	Precision/Recall								
			Convergence Criterion (%)								
			0.9			0.03			0.007		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20
45		6	0.50951	0.45691	0.42574	0.48357	0.44569	0.41613	0.48052	0.43944	0.40727

Table F.2.2: Result of experiment 1.2 when the scoring method is top 15 of 20 restarts.

No.	β	Number of Topics	Precision/Recall								
			Convergence Criterion (%)								
			0.9			0.03			0.007		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20
1	1	2	0.48842	0.46061	0.42970	0.45624	0.42835	0.41203	0.45700	0.43191	0.39572
2		3	0.48115	0.44367	0.40454	0.41913	0.38902	0.37699	0.42099	0.38256	0.35618
3		4	0.44559	0.41451	0.40295	0.42987	0.40503	0.38254	0.40618	0.38810	0.37289
4		5	0.46273	0.41870	0.41379	0.42896	0.39602	0.37148	0.42306	0.39342	0.35717
5		6	0.44492	0.40148	0.38539	0.41869	0.39383	0.37985	0.42713	0.39477	0.36687
6	0.95	2	0.50097	0.46664	0.43290	0.43502	0.42039	0.40181	0.43629	0.41670	0.40062
7		3	0.47725	0.44579	0.41747	0.43527	0.40937	0.39183	0.40788	0.37151	0.34790
8		4	0.47403	0.44740	0.41836	0.43749	0.40608	0.37224	0.41778	0.39753	0.37313
9		5	0.45638	0.42996	0.40772	0.42653	0.39086	0.36896	0.42159	0.38845	0.36937
10		6	0.45374	0.41989	0.39628	0.42267	0.40665	0.38379	0.40839	0.37685	0.36163
11	0.9	2	0.50377	0.46031	0.43148	0.44849	0.41403	0.40349	0.43659	0.41031	0.39172
12		3	0.50103	**0.47274**	0.43092	0.43844	0.41638	0.37961	0.41944	0.40100	0.38924
13		4	0.47474	0.44703	0.40260	0.42262	0.39488	0.35901	0.42402	0.39049	0.37528
14		5	0.44668	0.42721	0.41252	0.42789	0.41164	0.39251	0.42543	0.39375	0.37413
15		6	0.46498	0.42911	0.40124	0.43202	0.39906	0.38514	0.41996	0.39487	0.37970
16	0.85	2	0.50329	0.45051	0.42074	0.46341	0.43538	0.41584	0.44935	0.41764	0.40570
17		3	0.48857	0.44890	0.41108	0.44823	0.40965	0.39260	0.43851	0.40326	0.38285
18		4	0.49041	0.44523	0.42460	0.43118	0.41190	0.38436	0.43619	0.40505	0.38174
19		5	0.46675	0.43829	0.42006	0.43587	0.39849	0.38871	0.42641	0.40382	0.39520
20		6	0.46336	0.42208	0.40562	0.44137	0.41049	0.38852	0.44220	0.40228	0.39025
21	0.8	2	0.49760	0.45130	0.42888	0.46050	0.43730	0.41164	0.47390	0.44138	0.41682
22		3	0.48872	0.45905	0.42882	0.46706	0.41973	0.38859	0.45810	0.42746	0.39468

No.	β	Number of Topics	Precision/Recall								
			Convergence Criterion (%)								
			0.9			0.03			0.007		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20
23		4	0.49046	0.44994	0.43744	0.46619	0.42786	0.39316	0.46050	0.41590	0.39231
24		5	0.47233	0.43406	0.40743	0.44906	0.41107	0.39186	0.43852	0.40321	0.38829
25		6	0.46854	0.42848	0.39869	0.43438	0.39189	0.38142	0.45654	0.41868	0.38936
26	0.75	2	0.51084	0.46377	0.41980	0.48659	0.44137	0.40535	0.47616	0.44127	0.42761
27		3	0.49208	0.44775	0.42535	0.47166	0.43212	0.40416	0.46846	0.42451	0.41226
28		4	0.47849	0.43899	0.40861	0.44962	0.41877	0.41093	0.47563	0.42462	0.40663
29		5	0.47941	0.44579	0.42475	0.46034	0.42247	0.38967	0.45583	0.41929	0.39349
30		6	0.47859	0.43416	0.41824	0.45640	0.40694	0.39205	0.46995	0.42028	0.39075
31	0.7	2	**0.51165**	0.46765	**0.43747**	0.48898	0.45001	0.41555	0.48684	0.44776	0.43537
32		3	0.50841	0.47041	0.43292	0.48078	0.43939	0.41580	0.47857	0.43239	0.40741
33		4	0.48029	0.45238	0.42316	0.48690	0.43794	0.41165	0.47643	0.43750	0.41496
34		5	0.49490	0.45435	0.42116	0.46546	0.42454	0.40097	0.47455	0.43051	0.41143
35		6	0.49272	0.46073	0.43652	0.46652	0.43100	0.40963	0.47312	0.43315	0.40140
36	0.65	2	0.50406	0.46074	0.41846	0.49046	0.45163	0.42704	0.48834	0.44446	0.42247
37		3	0.50695	0.45408	0.42515	0.49200	0.45515	0.41727	0.47631	0.43511	0.42645
38		4	0.51047	0.46714	0.43314	0.48542	0.43890	0.39760	0.48699	0.44231	0.41496
39		5	0.50439	0.45748	0.42415	0.47678	0.43677	0.41466	0.49089	0.45129	0.41380
40		6	0.50043	0.44538	0.41987	0.48413	0.44219	0.40954	0.49058	0.44871	0.41135
41	0.6	2	0.51127	0.45909	0.43442	0.49348	0.46499	0.42881	0.49220	0.45151	0.42640
42		3	0.50654	0.46131	0.43318	0.49018	0.43044	0.41181	0.48419	0.44865	0.41945
43		4	0.50976	0.45608	0.42389	0.48071	0.43009	0.41226	0.48285	0.44146	0.41505
44		5	0.50622	0.45720	0.42473	0.48523	0.44412	0.41336	0.49190	0.43793	0.41772
45		6	0.50934	0.46211	0.42620	0.48187	0.43682	0.41718	0.48644	0.44518	0.42410

Table F.2.3: Result of experiment 1.2 when the scoring method is bottom 15 of 20 restarts.

No.	β	Number of Topics	Precision/Recall								
			Convergence Criterion (%)								
			0.9			0.03			0.007		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20
1	1	2	**0.52012**	0.46791	0.42549	0.47299	0.44758	0.42012	0.44571	0.42919	0.39868
2		3	0.48343	0.45411	0.42780	0.40582	0.38094	0.35190	0.41611	0.39387	0.37465
3		4	0.46186	0.43604	0.40588	0.42373	0.39159	0.36709	0.43286	0.41791	0.38565
4		5	0.47107	0.43389	0.40549	0.42327	0.39357	0.35718	0.41077	0.38359	0.35664
5		6	0.45616	0.42336	0.39177	0.44258	0.41514	0.38377	0.43302	0.39355	0.37587
6	0.95	2	0.49856	0.46693	0.42977	0.46392	0.42132	0.40553	0.43577	0.41734	0.40331
7		3	0.50197	0.46548	0.42749	0.43127	0.39929	0.37660	0.41484	0.39333	0.36397
8		4	0.48559	0.43933	0.41162	0.44660	0.41386	0.38534	0.40460	0.39420	0.36902
9		5	0.46970	0.43859	0.40766	0.42132	0.39368	0.37586	0.44309	0.42184	0.38414
10		6	0.47211	0.42454	0.39564	0.42050	0.39035	0.37045	0.42783	0.40061	0.37773
11	0.9	2	0.50184	0.45659	0.42591	0.45233	0.42232	0.41076	0.46087	0.42309	0.39963
12		3	0.47765	0.43366	0.41213	0.44749	0.42333	0.40465	0.42820	0.39619	0.37453
13		4	0.47272	0.43342	0.41156	0.44290	0.40591	0.37593	0.42593	0.39879	0.39204
14		5	0.46464	0.43886	0.40501	0.45902	0.43511	0.40529	0.42435	0.39488	0.37598
15		6	0.47464	0.42757	0.40185	0.44915	0.42052	0.38305	0.42143	0.40367	0.38939
16	0.85	2	0.50178	0.46393	0.42093	0.46974	0.43781	0.40661	0.46182	0.43989	0.42019
17		3	0.48937	0.45267	0.42903	0.44481	0.41412	0.38492	0.43803	0.40573	0.38760
18		4	0.49323	0.44251	0.42874	0.44147	0.41052	0.39472	0.43854	0.40406	0.38225
19		5	0.46841	0.44281	0.41444	0.44156	0.41644	0.38084	0.43446	0.39536	0.37741
20		6	0.46471	0.43325	0.40861	0.44566	0.41072	0.38225	0.43632	0.39615	0.37361
21	0.8	2	0.51036	0.46576	0.42630	0.48272	0.45268	0.42557	0.48372	0.43754	0.41019
22		3	0.49752	0.46255	0.43437	0.46890	0.43231	0.40408	0.45363	0.42789	0.40666

No.	β	Number of Topics	Precision/Recall								
			Convergence Criterion (%)								
			0.9			0.03			0.007		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20
23		4	0.49351	0.44675	0.41002	0.46196	0.41653	0.39267	0.45612	0.41559	0.38415
24		5	0.47313	0.43373	0.42383	0.45010	0.42497	0.40350	0.43752	0.40833	0.39072
25		6	0.48989	0.45843	0.41157	0.44898	0.42654	0.39088	0.45250	0.42334	0.40382
26		2	0.50278	0.45877	0.42513	0.47434	0.44443	0.42302	0.49260	0.44288	0.41001
27		3	0.50563	0.46515	0.42981	0.46423	0.43806	0.41632	0.46533	0.43435	0.39989
28	0.75	4	0.49262	0.46295	0.42347	0.46889	0.43223	0.40709	0.47939	0.44556	0.41748
29		5	0.48695	0.45373	0.41261	0.46306	0.42978	0.40422	0.45485	0.42135	0.39100
30		6	0.48549	0.44155	0.41856	0.46803	0.43011	0.39788	0.47288	0.42636	0.40796
31		2	0.50841	0.45792	0.42858	0.47950	0.44388	0.42117	0.49400	**0.47281**	0.43066
32		3	0.50270	0.46601	0.43051	0.47398	0.44766	0.40970	0.48928	0.44259	0.41906
33	0.7	4	0.49897	0.46384	0.43456	0.47874	0.44149	0.41358	0.46707	0.43041	0.41717
34		5	0.49894	0.46124	0.42282	0.47228	0.42780	0.40562	0.46747	0.44386	0.42442
35		6	0.48157	0.45300	0.40954	0.46622	0.43450	0.41411	0.47659	0.44148	0.42496
36		2	0.50701	0.45913	0.41784	0.49260	0.45626	0.43338	0.48887	0.44809	0.42502
37		3	0.50973	0.46299	0.42959	0.47850	0.43845	0.41142	0.48454	0.44313	0.41705
38	0.65	4	0.50848	0.45870	0.43214	0.48710	0.44928	0.42340	0.48519	0.44378	0.41980
39		5	0.51159	0.45595	0.42010	0.47620	0.44480	0.41292	0.48219	0.44452	0.42095
40		6	0.50855	0.45942	0.41979	0.48390	0.44438	0.41917	0.48141	0.44423	0.40979
41		2	0.50884	0.45869	0.42836	0.49742	0.45539	**0.44068**	0.49701	0.45802	0.42453
42		3	0.51037	0.45922	0.42972	0.48039	0.45017	0.42422	0.48428	0.44577	0.41904
43	0.6	4	0.51165	0.45693	0.42951	0.47574	0.43587	0.41696	0.48612	0.44554	0.41894
44		5	0.51043	0.45255	0.42268	0.48869	0.44174	0.40906	0.48953	0.44322	0.42109
45		6	0.50896	0.45567	0.42848	0.49004	0.45890	0.41599	0.48177	0.44848	0.41773

Table F.3: Averaged table (across topics) of F.2. It contains tables F.3.1, F.3.2 and F.3.3.

Table F.3.1: Averaged table (across topics) of F.2.1.

No.	β	Convergence Criterion (%)	Precision/Recall		
			Extraction Rate (%)		
			30	25	20
1	1	0.9	0.46885	0.43685	0.41006
2		0.03	0.43506	0.40710	0.38684
3		0.007	0.42227	0.40164	0.37876
4	0.95	0.9	0.47758	0.43779	0.41461
5		0.03	0.43657	0.40872	0.38501
6		0.007	0.42353	0.39334	0.37008
7	0.9	0.9	0.47959	0.44203	0.41554
8		0.03	0.43529	0.40763	0.38037
9		0.007	0.43586	0.40106	0.38372
10	0.85	0.9	0.49037	0.45133	0.42009
11		0.03	0.45161	0.41736	0.39126
12		0.007	0.44203	0.41460	0.39214
13	0.8	0.9	0.48281	0.44099	0.41600
14		0.03	0.46164	0.42688	0.39695
15		0.007	0.45600	0.42332	0.39609
16	0.75	0.9	0.49031	0.45054	0.41833
17		0.03	0.46631	0.43006	0.40316
18		0.007	0.46797	0.43378	0.40007
19	0.7	0.9	0.49355	0.45696	0.41929
20		0.03	0.47665	0.44452	0.41809
21		0.007	0.48231	0.44176	0.41377
22	0.65	0.9	**0.50879**	**0.45938**	0.42441
23		0.03	0.47868	0.44752	0.41776
24		0.007	0.48558	0.44492	0.41791
25	0.6	0.9	0.50844	0.45718	**0.42788**
26		0.03	0.48817	0.45267	0.42591
27		0.007	0.48824	0.44670	0.41523

Table F.3.2: Averaged table (across topics) of F.2.2.

No.	β	Convergence Criterion (%)	Precision/Recall		
			Extraction Rate (%)		
			30	25	20
1	1	0.9	0.46456	0.42779	0.40727
2		0.03	0.43058	0.40245	0.38458
3		0.007	0.42687	0.39815	0.36977
4	0.95	0.9	0.47248	0.44194	0.41455
5		0.03	0.43140	0.40667	0.38373
6		0.007	0.41838	0.39021	0.37053
7	0.9	0.9	0.47824	0.44728	0.41575
8		0.03	0.43389	0.40720	0.38395
9		0.007	0.42509	0.39808	0.38201
10	0.85	0.9	0.48248	0.44100	0.41642
11		0.03	0.44401	0.41318	0.39401
12		0.007	0.43853	0.40641	0.39115
13	0.8	0.9	0.48353	0.44457	0.42025
14		0.03	0.45544	0.41757	0.39333
15		0.007	0.45751	0.42133	0.39629
16	0.75	0.9	0.48788	0.44609	0.41935
17		0.03	0.46492	0.42433	0.40043
18		0.007	0.46921	0.42599	0.40615
19	0.7	0.9	0.49759	**0.46111**	**0.43024**
20		0.03	0.47773	0.43658	0.41072
21		0.007	0.47790	0.43626	0.41411
22	0.65	0.9	0.50526	0.45696	0.42415
23		0.03	0.48576	0.44493	0.41322
24		0.007	0.48662	0.44438	0.41780
25	0.6	0.9	**0.50862**	0.45916	0.42848
26		0.03	0.48630	0.44129	0.41668
27		0.007	0.48752	0.44494	0.42055

Table F.3.3: Averaged table (across topics) of F.2.3.

No.	β	Convergence Criteria (%)	Precision/Recall		
			Extraction Rate (%)		
			30	25	20
1	1	0.9	0.47853	0.44306	0.41129
2		0.03	0.43368	0.40576	0.37601
3		0.007	0.42769	0.40362	0.37830
4	0.95	0.9	0.48558	0.44697	0.41443
5		0.03	0.43672	0.40370	0.38276
6		0.007	0.42523	0.40546	0.37963
7	0.9	0.9	0.47830	0.43802	0.41129
8		0.03	0.45018	0.42144	0.39594
9		0.007	0.43216	0.40332	0.38632
10	0.85	0.9	0.48350	0.44703	0.42035
11		0.03	0.44865	0.41792	0.38987
12		0.007	0.44183	0.40824	0.38821
13	0.8	0.9	0.49288	0.45345	0.42122
14		0.03	0.46253	0.43061	0.40334
15		0.007	0.45670	0.42254	0.39911
16	0.75	0.9	0.49469	0.45643	0.42191
17		0.03	0.46771	0.43492	0.40971
18		0.007	0.47301	0.43410	0.40527
19	0.7	0.9	0.49812	**0.46040**	0.42520
20		0.03	0.47414	0.43906	0.41284
21		0.007	0.47888	0.44623	0.42325
22	0.65	0.9	0.50907	0.45924	0.42389
23		0.03	0.48366	0.44664	0.42006
24		0.007	0.48444	0.44475	0.41852
25	0.6	0.9	**0.51005**	0.45661	**0.42775**
26		0.03	0.48646	0.44841	0.42138
27		0.007	0.48774	0.44821	0.42026

Table F.4: Results of experiment 3. It contains tables F.4.1, F.4.2 and F.4.3.

Table F.4.1: Result of experiment 3 for

$Top\ 15\ of\ 20\ restarts, \beta = 0.7, and\ convergence\ criterion\ (\%) = 0.9.$

No.	Number of Keywords (%)	Number of Topics	Precision/Recall											
			α											
			2			3			4			5		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20	30	25	20
1	11	2	0.50917	0.45940	0.42913	0.51040	0.46182	0.42880	0.50873	0.45783	0.43393	0.51015	0.45463	0.42972
2		3	0.50478	0.45746	0.43760	0.49873	0.46055	0.43632	0.50476	0.45593	0.42242	0.50179	0.45581	0.41961
3		4	0.50299	0.45492	0.42862	0.50534	0.47058	0.43028	0.50872	0.46601	0.43220	0.50477	0.46461	0.42446
4		5	0.49389	0.46580	0.42868	0.50597	0.47344	0.42888	0.49790	0.46618	0.43224	0.49260	0.45163	0.42084
5		6	0.49641	0.45934	0.43898	0.48409	0.45940	0.43277	0.48890	0.45139	0.43069	0.49401	0.46314	0.42562
6	14	2	0.51453	0.45875	0.42617	0.50988	0.46247	0.42389	0.51531	0.45683	0.42938	0.50997	0.45891	0.42345
7		3	0.50145	0.45472	0.43354	0.50041	0.45018	0.43057	0.49900	0.45997	0.42857	0.49600	0.46719	0.42867
8		4	0.50235	0.46466	0.43119	0.49898	0.45282	0.43633	0.50900	0.45991	0.44006	0.50672	0.45783	0.43949
9		5	0.50515	0.46645	0.43543	0.50354	0.46220	0.44086	0.49138	0.46461	0.43453	0.49588	0.46222	0.43280
10		6	0.49689	0.47006	0.43194	0.49086	0.45684	0.42756	0.50394	0.47017	0.44634	0.49533	0.46747	0.44905
11	17	2	0.51089	0.45787	0.42308	0.50855	0.46219	0.42805	0.51572	0.45830	0.42780	0.51113	0.45921	0.42304
12		3	0.51105	0.46034	0.43384	0.50874	0.46004	0.43365	0.49974	0.45599	0.43021	0.50581	0.46824	0.44101
13		4	0.50992	0.47166	0.43813	0.51203	0.47062	0.43606	0.49684	0.45810	0.43583	0.50629	0.46647	0.43371
14		5	0.49192	0.46485	0.43548	0.50536	0.45882	0.43267	0.49177	0.46693	0.43032	0.50294	0.45750	0.43002
15		6	0.49825	0.46501	0.43622	0.49841	0.46863	0.44427	0.49168	0.45586	0.42850	0.49073	0.45720	0.42607
16	20	2	0.50621	0.45585	0.43031	0.51153	0.45982	0.42837	0.50315	0.45850	0.42492	0.51067	0.45412	0.42656
17		3	0.50662	0.46219	0.43763	0.50754	0.46264	0.43153	0.50855	0.46383	0.42615	0.50515	0.45710	0.43644
18		4	0.50712	0.46875	0.44519	0.51480	0.46162	0.43738	0.50477	0.47049	0.43541	0.51040	0.46596	0.44228
19		5	0.50339	0.46349	0.44312	0.51331	0.47653	0.43652	0.50314	0.46604	0.43251	0.49677	0.45091	0.42953

No.	Number of Keywords (%)	Number of Topics	Precision/Recall											
			α											
			2			3			4			5		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20	30	25	20
20		6	0.49880	0.46696	0.43771	0.49617	0.46689	0.44174	0.50417	0.46538	0.44046	0.49292	0.44740	0.42404
21		2	0.51113	0.45620	0.42870	0.50910	0.45104	0.42560	0.51276	0.46142	0.43122	0.50891	0.45638	0.42953
22		3	0.51306	0.46406	0.43213	0.50393	0.45828	0.44067	0.50510	0.46121	0.44035	**0.51644**	0.46852	0.43963
23	23	4	0.50704	0.47806	0.44209	0.51384	0.47704	0.44168	0.50967	0.46904	0.43372	0.50874	0.46829	0.43692
24		5	0.49733	0.46535	0.44236	0.50839	0.47538	0.44588	0.50450	0.46233	0.43859	0.48546	0.45646	0.42912
25		6	0.49283	0.46124	0.43276	0.49612	0.45899	0.44078	0.48994	0.45646	0.43598	0.49317	0.45878	0.43287
26		2	0.50769	0.45781	0.42871	0.50723	0.45356	0.42725	0.51074	0.45602	0.42556	0.51037	0.45693	0.43329
27		3	0.49311	0.45224	0.42758	0.51388	0.45776	0.43656	0.50616	0.46233	0.43380	0.50338	0.46258	0.43058
28	26	4	0.50698	0.46690	0.44370	0.51402	0.46881	0.44856	0.50914	0.47297	0.43971	0.50713	0.47089	0.43238
29		5	0.49619	0.46709	0.44427	0.49864	0.46605	0.44746	0.50013	0.46705	0.43647	0.49987	0.45421	0.42940
30		6	0.50068	0.45994	0.44159	0.49718	0.46817	0.43457	0.50324	**0.48499**	0.45472	0.49977	0.46239	0.42785
31		2	0.50662	0.46224	0.43134	0.51143	0.46160	0.42313	0.50622	0.46444	0.42278	0.51345	0.46851	0.42533
32		3	0.51030	0.45688	0.43721	0.50580	0.46061	0.43707	0.50682	0.46328	0.44190	0.51260	0.46462	0.42334
33	29	4	0.51465	0.47147	0.43323	0.51627	0.46989	0.44416	0.49710	0.46612	0.42966	0.50930	0.47173	0.43549
34		5	0.49735	0.45735	0.43737	0.49577	0.46174	0.43951	0.50039	0.46042	0.42555	0.49293	0.45938	0.44316
35		6	0.49985	0.46935	0.44735	0.49305	0.46921	0.43841	0.48942	0.46179	**0.45511**	0.49547	0.46351	0.43187
36		2	0.51153	0.46489	0.42561	0.50612	0.45984	0.43069	0.51193	0.45779	0.42956	0.51425	0.46048	0.41870
37		3	0.50905	0.46639	0.43784	0.51384	0.47334	0.43521	0.50352	0.45772	0.44287	0.51052	0.46332	0.44030
38	32	4	0.49427	0.45232	0.42118	0.50410	0.46502	0.43710	0.49799	0.46107	0.43490	0.50574	0.45834	0.43913
39		5	0.50390	0.46411	0.44160	0.49423	0.46343	0.44256	0.49893	0.45676	0.44098	0.48566	0.44792	0.42874
40		6	0.50156	0.46830	0.44383	0.48501	0.45285	0.43187	0.49533	0.45795	0.44717	0.49503	0.45431	0.43279

Table F.4.2: Result of experiment 3 for

$Bottom\ 10\ of\ 20\ restarts, \beta = 0.65, and\ convergence\ criterion\ (\%) = 0.9.$

No.	Number of Keywords (%)	Number of Topics	Precision/Recall											
			α											
			2			3			4			5		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20	30	25	20
1	11	2	0.51092	0.45758	0.42718	0.50898	0.45699	0.42755	0.51038	0.45766	0.42724	0.51606	0.46156	0.43081
2		3	0.51187	0.46224	0.42720	0.51099	0.46023	0.42733	0.50945	0.45945	0.42906	0.51114	0.45946	0.43032
3		4	0.51014	0.46321	0.42718	0.50983	0.45960	0.42564	0.51433	0.45698	0.42721	0.51032	0.46156	0.42838
4		5	0.50386	0.45719	0.42324	0.50299	0.45331	0.41867	0.51014	0.46283	0.43067	0.51015	0.45894	0.42162
5		6	0.50807	0.45500	0.42786	0.50831	0.46385	**0.43623**	0.50669	0.45556	0.42425	0.50331	0.45610	0.42910
6	14	2	0.50958	0.46097	0.42646	0.50996	0.45778	0.42505	0.50884	0.45884	0.42439	0.50639	0.45406	0.42490
7		3	0.51526	0.46063	0.42548	0.51217	0.46224	0.42466	0.51338	0.46425	0.42893	0.51116	0.45843	0.43047
8		4	0.50877	0.45704	0.42277	0.51138	0.45940	0.42675	0.51094	0.46232	0.42678	0.51111	0.45753	0.42536
9		5	0.51313	0.45932	0.42124	0.50871	0.46019	0.42789	0.50674	0.45425	0.42293	0.51276	0.45898	0.42804
10		6	0.50204	0.45534	0.42177	0.50494	0.45679	0.42086	0.49960	0.46270	0.42342	0.50969	0.46213	0.42105
11	17	2	0.51180	0.46234	0.42675	0.50861	0.45815	0.42698	0.50829	0.45742	0.41980	0.51156	0.46239	0.43318
12		3	0.51067	0.45926	0.43236	0.51533	0.46101	0.42951	0.50829	0.46321	0.43090	0.51132	0.45824	0.42301
13		4	0.51217	0.46063	0.43240	0.50988	0.45926	0.42587	0.51211	0.46016	0.42196	0.50718	0.45940	0.42485
14		5	0.51112	0.45849	0.43054	0.50562	0.46123	0.42450	0.50923	0.45687	0.41869	0.50718	0.45724	0.42055
15		6	0.50632	0.45681	0.43062	0.50214	0.46182	0.42410	0.50815	0.45601	0.43207	0.50961	0.46087	0.42967
16	20	2	0.51180	0.46000	0.42419	0.50821	0.45752	0.42948	0.50774	0.45603	0.42562	**0.51669**	0.45992	0.41828
17		3	0.50891	0.45839	0.42549	0.51191	0.46056	0.42408	0.51204	0.46079	0.42382	0.51123	0.46301	0.42818
18		4	0.50921	0.45994	0.42286	0.50786	0.45820	0.42771	0.51032	0.45371	0.42382	0.50871	0.46326	0.42787
19		5	0.50651	0.45517	0.42281	0.51056	0.46478	0.41727	0.50697	0.46000	0.42203	0.51101	0.45606	0.43110
20		6	0.50500	0.45671	0.42688	0.50537	0.46317	0.42506	0.51369	0.46522	0.42470	0.50517	0.45614	0.42787
21	23	2	0.50710	0.45823	0.42629	0.51302	0.46409	0.42490	0.50883	0.45562	0.42683	0.51419	0.45753	0.43189

No.	Number of Keywords (%)	Number of Topics	Precision/Recall											
			α											
			2			3			4			5		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20	30	25	20
22		3	0.51135	0.45593	0.42481	0.50778	0.45918	0.42058	0.51175	0.45801	0.42409	0.50952	0.45986	0.42379
23		4	0.50959	0.46278	0.43022	0.51043	0.45432	0.42925	0.51481	0.45954	0.41758	0.50847	0.45949	0.42870
24		5	0.50922	0.45776	0.42625	0.50565	0.45575	0.41748	0.50868	0.45214	0.42179	0.51258	0.45633	0.43092
25		6	0.50416	0.45860	0.42464	0.50447	0.44561	0.41930	0.50280	0.45605	0.42739	0.50714	**0.46560**	0.42702
26		2	0.50996	0.45702	0.42274	0.51154	0.46051	0.41851	0.50618	0.45242	0.42629	0.50959	0.46071	0.43191
27		3	0.51324	0.46264	0.42786	0.50835	0.45488	0.42471	0.51169	0.45912	0.43187	0.51105	0.45793	0.42623
28	26	4	0.51272	0.46037	0.42732	0.50994	0.45668	0.42382	0.51242	0.46148	0.43113	0.51407	0.45432	0.41789
29		5	0.50993	0.45753	0.42604	0.51109	0.45542	0.42531	0.50643	0.45641	0.42205	0.50907	0.45859	0.41970
30		6	0.50983	0.45559	0.42032	0.50596	0.45567	0.42632	0.50097	0.45789	0.42392	0.51014	0.45537	0.42516
31		2	0.51241	0.46085	0.42834	0.51340	0.45954	0.42513	0.51152	0.45759	0.42019	0.51512	0.45959	0.43225
32		3	0.51218	0.45610	0.42360	0.50947	0.46093	0.42466	0.50841	0.45744	0.42666	0.50764	0.46188	0.42417
33	29	4	0.50958	0.45841	0.42340	0.51119	0.45926	0.42357	0.51398	0.46232	0.42271	0.50946	0.45952	0.42870
34		5	0.50649	0.45939	0.42638	0.50730	0.46170	0.42591	0.50884	0.46449	0.41793	0.50756	0.46330	0.42671
35		6	0.50065	0.45489	0.42224	0.50756	0.45774	0.42165	0.49887	0.45082	0.41653	0.49995	0.45548	0.41687
36		2	0.51087	0.45540	0.42478	0.50689	0.45736	0.42456	0.51397	0.46014	0.42716	0.50928	0.46091	0.41972
37		3	0.51094	0.46164	0.43014	0.50977	0.45994	0.42312	0.51123	0.46175	0.42451	0.50842	0.46411	0.42771
38	32	4	0.51306	0.46224	0.42608	0.50818	0.45932	0.42388	0.50888	0.46231	0.41874	0.50965	0.46046	0.41951
39		5	0.50569	0.45119	0.41877	0.50814	0.45764	0.41827	0.50983	0.46347	0.42308	0.51424	0.46049	0.42165
40		6	0.51429	0.45622	0.42177	0.50926	0.46276	0.42562	0.50108	0.46159	0.42892	0.51455	0.46336	0.42810

Table F.4.3: Result of experiment 3 for

$Bottom\ 15\ of\ 20\ restarts, \beta = 0.6, and\ convergence\ criterion\ (\%) = 0.9.$

No.	Number of Keywords (%)	Number of Topics	Precision/Recall											
			α											
			2			3			4			5		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20	30	25	20
1	11	2	0.51000	0.45712	0.42273	0.51146	0.46017	0.42754	0.50945	0.46071	0.42675	0.50938	0.46153	0.42736
2		3	0.51276	0.46363	0.42569	0.50829	0.45925	0.42412	0.50921	0.46148	0.42610	0.51193	0.45883	0.42598
3		4	0.51145	0.45994	0.42939	0.51062	0.45753	0.42507	0.51068	0.45843	0.42502	0.51067	0.46162	0.42831
4		5	0.51025	0.45753	0.42692	0.51100	0.45843	0.42909	0.51203	0.45759	0.42795	0.51040	0.45670	0.42257
5		6	0.50855	0.46056	0.42831	0.50859	0.45759	0.42721	0.50780	0.45962	0.42754	0.50766	0.45704	0.42925
6	14	2	0.51573	0.45940	0.42479	0.51135	0.45851	0.42374	0.51217	0.46008	0.42011	0.51062	0.45905	0.42140
7		3	0.50976	0.45906	0.42521	0.51276	0.45746	0.42397	0.51043	0.46147	0.43004	0.50994	0.46085	0.42601
8		4	0.51376	0.46202	0.42744	0.51171	0.45966	0.42646	0.51192	0.45903	0.42369	0.50827	0.45813	0.42953
9		5	0.51079	0.45940	0.42832	0.51203	0.46017	0.42656	0.50965	0.45815	0.42860	0.51094	0.45877	0.42430
10		6	0.51178	0.46141	0.42539	0.51598	0.46085	0.42567	0.51242	0.46023	0.42773	0.51094	0.46133	0.42412
11	17	2	0.50859	0.45875	0.41946	0.50970	0.45982	0.42744	0.51179	0.46286	0.42308	0.51119	0.45980	0.42459
12		3	0.51407	0.46093	0.42760	0.51113	0.46051	0.42536	0.51381	0.46125	0.42459	0.50938	**0.46403**	0.42644
13		4	0.51348	0.46000	0.42420	0.51075	0.46133	0.42767	0.50938	0.45772	0.42462	0.51067	0.45906	0.42415
14		5	0.50915	0.46133	0.42397	0.50891	0.45954	0.42629	0.51129	0.46079	0.42678	0.50952	0.46017	0.42466
15		6	0.50916	0.45726	0.42547	0.51149	0.45871	0.42490	0.51146	0.46017	0.42582	0.51043	0.46280	0.42814
16	20	2	0.50902	0.45940	0.42646	0.50786	0.45935	0.42701	0.51022	0.46042	0.42723	0.51010	0.46017	0.42728
17		3	0.50938	0.46084	0.42389	0.51191	0.45759	0.42428	0.50915	0.45710	0.42814	0.51340	0.46224	0.42686
18		4	0.50946	0.45918	0.42561	0.51161	0.46085	0.42755	0.50754	0.45875	0.42392	0.51018	0.46009	0.42548
19		5	0.50767	0.45932	0.42721	0.51067	0.45778	0.42539	0.50908	0.46091	0.42937	0.50970	0.45857	0.42531
20		6	0.51363	0.46079	0.42598	0.51111	0.45780	0.42721	0.51099	0.45918	0.42448	0.51067	0.45726	**0.43141**
21	23	2	0.50756	0.45655	0.42516	**0.51600**	0.46295	0.43055	0.51081	0.45996	0.42597	0.51105	0.45434	0.42528

No.	Number of Keywords (%)	Number of Topics	Precision/Recall											
			α											
			2			3			4			5		
			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)			Extraction Rate (%)		
			30	25	20	30	25	20	30	25	20	30	25	20
22		3	0.50892	0.45898	0.42706	0.50803	0.45945	0.42317	0.51302	0.45986	0.42335	0.50915	0.45652	0.42371
23		4	0.50978	0.45732	0.42678	0.51105	0.45509	0.42989	0.51038	0.46071	0.42132	0.50926	0.45648	0.42605
24		5	0.50962	0.45925	0.42402	0.51487	0.46085	0.42659	0.51056	0.46056	0.42716	0.50921	0.45884	0.42675
25		6	0.50970	0.45801	0.42448	0.50823	0.45932	0.42744	0.50908	0.45801	0.42539	0.51246	0.46042	0.42865
26	26	2	0.50785	0.45706	0.42584	0.50855	0.45834	0.43045	0.51327	0.45820	0.42917	0.51184	0.45601	0.42108
27		3	0.50753	0.45792	0.42551	0.51049	0.46155	0.42629	0.51050	0.46293	0.41948	0.50970	0.45795	0.42710
28		4	0.51026	0.45309	0.42428	0.51043	0.46056	0.42345	0.50962	0.45778	0.42698	0.51161	0.45762	0.42531
29		5	0.51111	0.46050	0.42624	0.50959	0.45889	0.42358	0.50867	0.45968	0.42263	0.51049	0.46210	0.42659
30		6	0.50908	0.45986	0.42142	0.50767	0.46042	0.42412	0.51302	0.45917	0.42132	0.51061	0.45871	0.42363
31	29	2	0.51144	0.45682	0.42490	0.51240	0.45599	0.42296	0.50619	0.45690	0.42942	0.51058	0.45709	0.41993
32		3	0.50656	0.45599	0.42528	0.50810	0.45807	0.42700	0.51093	0.46016	0.42466	0.51375	0.45670	0.42414
33		4	0.51129	0.45925	0.42207	0.51008	0.45571	0.42652	0.51037	0.46168	0.42590	0.50976	0.45925	0.41957
34		5	0.51204	0.46002	0.42266	0.51106	0.45766	0.42190	0.50724	0.45974	0.42479	0.50994	0.45563	0.42575
35		6	0.51087	0.46002	0.42328	0.51438	0.45871	0.42794	0.50855	0.45655	0.43138	0.51026	0.45925	0.42675
36	32	2	0.51153	0.46221	0.42659	0.50803	0.45313	0.42632	0.51173	0.45871	0.42044	0.50908	0.46063	0.42999
37		3	0.51192	0.45591	0.42554	0.51280	0.46042	0.43064	0.51342	0.45792	0.42062	0.50782	0.45726	0.42662
38		4	0.51093	0.46293	0.42281	0.50845	0.45759	0.42798	0.51407	0.45815	0.42554	0.51240	0.45974	0.42610
39		5	0.50607	0.45820	0.42505	0.50835	0.46121	0.42420	0.50859	0.45662	0.42633	0.51069	0.46016	0.42605
40		6	0.50635	0.45496	0.42818	0.50847	0.45724	0.42767	0.51248	0.46301	0.42108	0.50915	0.45940	0.42425

Appendix G. User Interface of the Summarizer

We have implemented a prototype summarizer whose user interface is shown below in figure G.1. The user needs to input the document without its title. The summarizer has three options:

- Small-sized summary (summary at 20% extraction rate)
- Medium-sized summary (summary at 25% extraction rate)
- Large-sized summary (summary at 30% extraction rate)

The summary at each extraction rate is generated using the configurations of algorithms that have performed the best at the given extraction rate according to the results of our experiments.

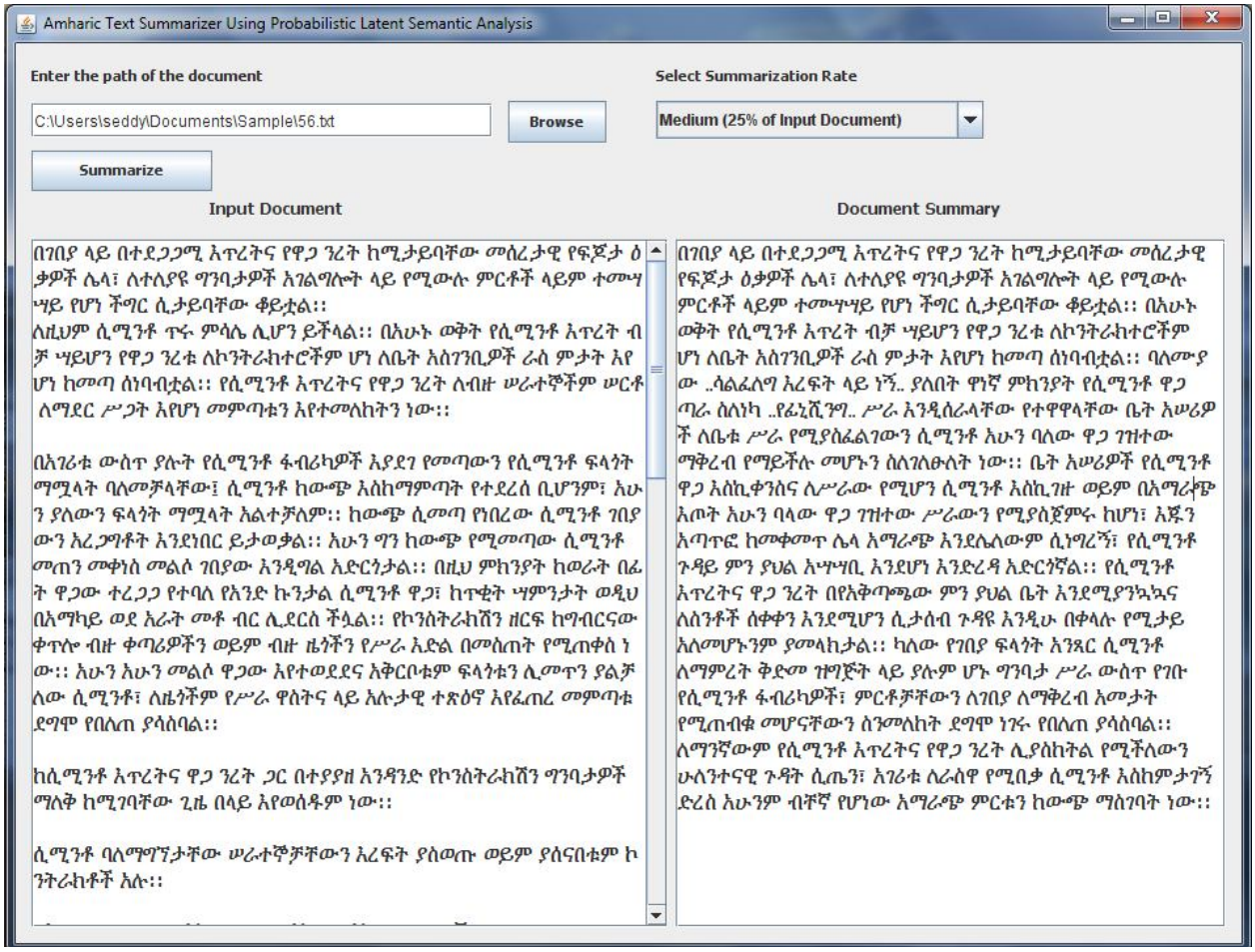

Figure G.1: User interface of the summarizer. User interface is adapted from (Melese 2009).

Appendix H. Sample Generated Summary

This appendix gives generated summary for a sample document at 25% extraction rate. The summary is generated using the configuration of the algorithm that has performed the best at 25% extraction rate according to the results of our experiments.

Input Document

ሲሚንቶ እንደገና…

በገበያ ላይ በተደጋጋሚ እጥረትና የዋጋ ንረት ከሚታይባቸው መሰረታዊ የፍጆታ ዕቃዎች ሌላ፣ ለተለያዩ ግንባታዎች አገልግሎት ላይ የሚውሉ ምርቶች ላይም ተመሣሣይ የሆነ ችግር ሲታይባቸው ቆይቷል፡፡ ለዚህም ሲሚንቶ ጥሩ ምሳሌ ሊሆን ይችላል፡፡ በአሁኑ ወቅት የሲሚንቶ እጥረት ብቻ ሣይሆን የዋጋ ንረቱ ለኮንትራክተሮችም ሆነ ለቤት አስገንቢዎች ራስ ምታት እየሆነ ከመጣ ሰነባብቷል፡፡ የሲሚንቶ እጥረትና የዋጋ ንረት ለብዙ ሠራተኞችም ሠርቶ ለማደር ሥጋት እየሆነ መምጣቱን እየተመለከትን ነው፡፡

በአገሪቱ ውስጥ ያሉት የሲሚንቶ ፋብሪካዎች እያደገ የመጣውን የሲሚንቶ ፍላጎት ማሟላት ባለመቻላቸው፤ ሲሚንቶ ከውጭ እስከማምጣት የተደረሰ ቢሆንም፣ አሁን ያለውን ፍላጎት ማሟላት አልተቻለም፡፡ ከውጭ ሲመጣ የነበረው ሲሚንቶ ገበያውን አረጋግቶት እንደነበር ይታወቃል፡፡ አሁን ግን ከውጭ የሚመጣው ሲሚንቶ መጠን መቀነስ መልሶ ገበያው እንዲግል አድርጎታል፡፡ በዚህ ምክንያት ከወራት በፊት ዋጋው ተረጋጋ የተባለ የአንድ ኩንታል ሲሚንቶ ዋጋ፣ ከጥቂት ሣምንታት ወዲህ በአማካይ ወደ አራት መቶ ብር ሊደርስ ችሏል፡፡ የኮንስትራክሽን ዘርፍ ከግብርናው ቀጥሎ ብዙ ቀጣሪዎችን ወይም ብዙ ዜጎችን የሥራ እድል በመስጠት የሚጠቀስ ነው፡፡ አሁን አሁን መልሶ ዋጋው እየተወደደና አቅርቦቱም ፍላጎቱን ሊመጥን ያልቻለው ሲሚንቶ፣ ለዜጎችም የሥራ ዋስትና ላይ አሉታዊ ተጽዕኖ እየፈጠረ መምጣቱ ደግሞ የበለጠ ያሳስባል፡፡

ከሲሚንቶ እጥረትና ዋጋ ንረት ጋር በተያያዘ አንዳንድ የኮንስትራክሽን ግንባታዎች ማለቅ ከሚገባቸው ጊዜ በላይ እየወሰዱም ነው፡፡

ሲሚንቶ ባለማግኘታቸው ሠራተኞቻቸውን እረፍት ያስወጡ ወይም ያሰናበቱም ኮንትራክቶች አሉ፡፡

በቅርቡ በግሉ አነስተኛ የኮንስትራክሽን "ፊኒሺንግ" ሥራዎችን የሚሠራ አንድ ሰው አግኝቼ ስለ ሥራው ስጠይቀው፣ "ሰሞኑን በግድ ለራሴ እረፍት ሰጥቻለሁ" ነበር ያለኝ፡፡

ባለሙያው ..ሳልፈለግ እረፍት ላይ ነኝ.. ያለበት ዋነኛ ምክንያት የሲሚንቶ ዋጋ ጣራ ስለነካ ..የፊኒሺንግ.. ሥራ እንዲሰራላቸው የተዋዋላቸው ቤት አሠሪዎች ለቤቱ ሥራ የሚያስፈልገውን ሲሚንቶ አሁን ባለው ዋጋ ገዝተው ማቅረብ የማይችሉ መሆኑን ስለገለፁለት ነው፡፡

እንደባለሙያው ገለጻ፣ ይህ ገጠመኙ አንድ ቦታ ብቻ ሳይሆን ተመሣሣይ ሥራ ለመስራት ውል ገብቶባቸው የነበሩ ሁለትና ሦስት ቤት አስገንቢዎች ጋር ያጋጠመው በመሆኑ ሥራውን ለማቆም መገደዱን ገልጾልኛል፡፡ አብረውት ሲሠሩ የነበሩትንም ሦስት ሠራተኞች እንዲያርፉ አድርጎ ያለሥራ የአንድ ሣምንት ደሞዛቸውን መክፈሉንም ነገረኝ፡፡

ቤት አሠሪዎች የሲሚንቶ ዋጋ እስኪቀንስና ለሥራው የሚሆን ሲሚንቶ እስኪገዙ ወይም በአማራጭ እጦት አሁን ባላው ዋጋ ገዝተው ሥራውን የሚያስጀምሩ ከሆነ፣ እጁን አጣጥፎ ከመቀመጥ ሌላ አማራጭ እንደሌለውም ሲነግረኝ፣ የሲሚንቶ ጉዳይ ምን ያህል አሣሣቢ እንደሆነ እንድረዳ አድርጎኛል።

የሲሚንቶ እጥረትና ዋጋ ንረት በየአቅጣጫው ምን ያህል ቤት እንደሚያንኳኳና ለስንቶች ሰቀቀን እንደሚሆን ሲታሰብ ጉዳዩ እንዲሁ በቀላሉ የሚታይ አለመሆኑንም ያመላክታል።

በአሁኑ ወቅት የሚካሄዱ ግንባታዎች ብዛት አንፃር ችግሩ ነገም፣ ተነገ ወዲያም ሊቀጥል ይችላል።

ካለው የገበያ ፍላጎት አንጻር ሲሚንቶ ለማምረት ቅድመ ዝግጅት ላይ ያሉም ሆኑ ግንባታ ሥራ ውስጥ የገቡ የሲሚንቶ ፋብሪካዎች፣ ምርቶቻቸውን ለገበያ ለማቅረብ አመታት የሚጠብቁ መሆናቸውን ስንመለከት ደግሞ ነገሩ የበለጠ ያሳስባል። ማን ያውቃል እነዚህም የሲሚንቶ ፋብሪካዎች በሲሚንቶ እጥረት ሳቢያ የግንባታ ማጠናቀቂያ ጊዜያቸው ሊጓተት ይችልም ይሆናል።

ለማንኛውም የሲሚንቶ እጥረትና የዋጋ ንረት ሊያስከትል የሚችለውን ሁለንተናዊ ጉዳት ሲጤን፣ አገሪቱ ለራስዋ የሚበቃ ሲሚንቶ እስከምታገኝ ድረስ አሁንም ብቸኛ የሆነው አማራጭ ምርቱን ከውጭ ማስገባት ነው።

ይህ እንዲሆንም መንግሥት ቅድሚያ ሃላፊነት ያለበትና ጉዳዩ በቀጥታ የሚመለከታቸው ኮንትራክተሮችና የንግዱ ህብረተሰብ ምርቱን በማስገባት ለራሳቸው የሚጠቀሙበትና እንዲሁም በአግባቡ ለተጠቃሚዎች ሊያደርሱ የሚችሉበት ጊዜያዊ መመሪያ ተቀርጾ ሥራ ላይ መዋል ይገባዋል።

ይህን ማድረግ አሁን ያለውን ዋጋ በማረጋጋት የራሱ የሆነ ሚና ይኖረዋል። ችግሩ አገር አቀፍ በመሆኑም መንግሥት እንዲገባ የሚፈቅደው የሲሚንቶ ምርት በትክክል ለሚገነቡ አካላት ስለመድረሱ የሚያረጋግጥ ቁጥጥር ማድረግም ይጠበቅበታል። ብዙ ጊዜ በሲሚንቶ ግብይት ዙርያ በትክክል ሲሚንቶ የሚያስፈልጋቸው ድርጅቶችና ግለሰቦች ያልተገባ ድርጊት እየፈጸሙ የሚታይ በመሆኑ፣ ለተወሰነ ጥቅም የሚሯሯጠውንና ትክክለኛውን ገንቢ ለመለየት የሚያስችል አሠራር መተግበር ይኖርበታል። ይህ ካልሆነ ሰው ሰራሽ ክስተቶች የሚያስከትሉት ችግር ሁሌም የሲሚንቶ ገበያ እንዳይረጋጋ ያደርገዋል።

Ideal Summary 1

በገበያ ላይ በተደጋጋሚ እጥረትና የዋጋ ንረት ከሚታይባቸው መሰረታዊ የፍጆታ ዕቃዎች ሌላ፣ ለተለያዩ ግንባታዎች አገልግሎት ላይ የሚውሉ ምርቶች ላይም ተመሣሣይ የሆነ ችግር ሲታይባቸው ቆይቷል። በአሁኑ ወቅት የሲሚንቶ እጥረት ብቻ ሣይሆን የዋጋ ንረቱ ለኮንትራክተሮችም ሆነ ለቤት አስገንቢዎች ራስ ምታት እየሆነ ከመጣ ሰነባብቷል። የሲሚንቶ እጥረትና የዋጋ ንረት ለብዙ ሠራተኞችም ሠርቶ ለማደር ሥጋት እየሆነ መምጣቱን እየተመለከትን ነው። የሲሚንቶ እጥረትና ዋጋ ንረት በየአቅጣጫው ምን ያህል ቤት እንደሚያንኳኳና ለስንቶች ሰቀቀን እንደሚሆን ሲታሰብ ጉዳዩ እንዲሁ በቀላሉ የሚታይ አለመሆኑንም ያመላክታል። ካለው የገበያ ፍላጎት አንጻር ሲሚንቶ ለማምረት ቅድመ ዝግጅት ላይ ያሉም ሆኑ ግንባታ ሥራ ውስጥ የገቡ የሲሚንቶ ፋብሪካዎች፣ ምርቶቻቸውን ለገበያ ለማቅረብ አመታት የሚጠብቁ መሆናቸውን ስንመለከት ደግሞ ነገሩ የበለጠ ያሳስባል። ለማንኛውም የሲሚንቶ እጥረትና የዋጋ ንረት ሊያስከትል የሚችለውን ሁለንተናዊ ጉዳት ሲጤን፣ አገሪቱ ለራስዋ የሚበቃ ሲሚንቶ እስከምታገኝ ድረስ አሁንም ብቸኛ የሆነው

አማራጭ ምርቱን ከውጭ ማስገባት ነው፡፡ ይህ እንዲሆንም መንግሥት ቅድሚያ ሃላፊነት ያለበትና ጉዳዩ በቀጥታ የሚመለከታቸው ኮንትራክተሮችና የንግዱ ህብረተሰብ ምርቱን በማስገባት ለራሳቸው የሚጠቀሙበትና እንዲሁም በአግባቡ ለተጠቃሚዎች ሊያደርሱ የሚችሉበት ጊዜያዊ መመሪያ ተቀርጾ ሥራ ላይ መዋል ይገባዋል፡፡

Ideal Summary 2

በገበያ ላይ በተደጋጋሚ እጥረትና የዋጋ ንረት ከሚታይባቸው መሰረታዊ የፍጆታ ዕቃዎች ሌላ፣ ለተለያዩ ግንባታዎች አገልግሎት ላይ የሚውሉ ምርቶች ላይም ተመሣሣይ የሆነ ችግር ሲታይባቸው ቆይቷል፡፡ በአሁኑ ወቅት የሲሚንቶ እጥረት ብቻ ሣይሆን የዋጋ ንረቱ ለኮንትራክተሮችም ሆነ ለቤት አስገንቢዎች ራስ ምታት እየሆነ ከመጣ ሰነባብቷል፡፡ የሲሚንቶ እጥረትና የዋጋ ንረት ለብዙ ሠራተኞችም ሠርቶ ለማደር ሥጋት እየሆነ መምጣቱን እየተመለከትን ነው፡፡ በዚህ ምክንያት ከወራት በፊት ዋጋው ተረጋጋ የተባለ የአንድ ኩንታል ሲሚንቶ ዋጋ፣ ከጥቂት ሣምንታት ወዲህ በአማካይ ወደ አራት መቶ ብር ሊደርስ ችሏል፡፡ ፡ ለማንኛውም የሲሚንቶ እጥረትና የዋጋ ንረት ሊያስከትል የሚችለውን ሁለንተናዊ ጉዳት ሲጤን፣ አገሪቱ ለራስዋ የሚበቃ ሲሚንቶ እስከምታገኝ ድረስ አሁንም ብቸኛ የሆነው አማራጭ ምርቱን ከውጭ ማስገባት ነው፡፡ ይህ እንዲሆንም መንግሥት ቅድሚያ ሃላፊነት ያለበትና ጉዳዩ በቀጥታ የሚመለከታቸው ኮንትራክተሮችና የንግዱ ህብረተሰብ ምርቱን በማስገባት ለራሳቸው የሚጠቀሙበትና እንዲሁም በአግባቡ ለተጠቃሚዎች ሊያደርሱ የሚችሉበት ጊዜያዊ መመሪያ ተቀርጾ ሥራ ላይ መዋል ይገባዋል፡፡ ችግሩ አገር አቀፍ በመሆኑም መንግሥት እንዲገባ የሚፈቅደው የሲሚንቶ ምርት በትክክል ለሚገነቡ አካላት ስለመድረሱ የሚያረጋግጥ ቁጥጥር ማድረግም ይጠበቅበታል፡፡

Ideal Summary 3

በአሁኑ ወቅት የሲሚንቶ እጥረት ብቻ ሣይሆን የዋጋ ንረቱ ለኮንትራክተሮችም ሆነ ለቤት አስገንቢዎች ራስ ምታት እየሆነ ከመጣ ሰነባብቷል፡፡የሲሚንቶ እጥረትና የዋጋ ንረት ለብዙ ሠራተኞችም ሠርቶ ለማደር ሥጋት እየሆነ መምጣቱን እየተመለከትን ነው፡፡ በአገሪቱ ውስጥ ያሉት የሲሚንቶ ፋብሪካዎች እያደገ የመጣውን የሲሚንቶ ፍላጎት ማሟላት ባለመቻላቸው፤ ሲሚንቶ ከውጭ እስከማምጣት የተደረሰ ቢሆንም፣ አሁን ያለውን ፍላጎት ማሟላት አልተቻለም፡፡ ከሲሚንቶ እጥረትና ዋጋ ንረት ጋር በተያያዘ አንዳንድ የኮንስትራክሽን ግንባታዎች ማለቅ ከሚገባቸው ጊዜ በላይ እየወሰዱም ነው፡፡ በአሁኑ ወቅት የሚካሄዱ ግንባታዎች ብዛት አንፃር ችግሩ ነገም፣ ተነገ ወዲያም ሊቀጥል ይችላል፡፡ ለማንኛውም የሲሚንቶ እጥረትና የዋጋ ንረት ሊያስከትል የሚችለውን ሁለንተናዊ ጉዳት ሲጤን፣ አገሪቱ ለራስዋ የሚበቃ ሲሚንቶ እስከምታገኝ ድረስ አሁንም ብቸኛ የሆነው አማራጭ ምርቱን ከውጭ ማስገባት ነው፡፡ ይህ እንዲሆንም መንግሥት ቅድሚያ ሃላፊነት ያለበትና ጉዳዩ በቀጥታ የሚመለከታቸው ኮንትራክተሮችና የንግዱ ህብረተሰብ ምርቱን በማስገባት ለራሳቸው የሚጠቀሙበትና እንዲሁም በአግባቡ ለተጠቃሚዎች ሊያደርሱ የሚችሉበት ጊዜያዊ መመሪያ ተቀርጾ ሥራ ላይ መዋል ይገባዋል፡፡

System Summary

በገበያ ላይ በተደጋጋሚ እጥረትና የዋጋ ንረት ከሚታይባቸው መሰረታዊ የፍጆታ ዕቃዎች ሌላ፣ ለተለያዩ ግንባታዎች አገልግሎት ላይ የሚውሉ ምርቶች ላይም ተመሣሣይ የሆነ ችግር ሲታይባቸው ቆይቷል፡፡ በአሁኑ ወቅት የሲሚንቶ እጥረት ብቻ ሣይሆን የዋጋ ንረቱ ለኮንትራክተሮችም ሆነ ለቤት አስገንቢዎች ራስ ምታት እየሆነ ከመጣ ሰነባብቷል፡፡ ባለሙያው ..ሳልፈለግ እረፍት ላይ ነኝ.. ያለበት ዋነኛ ምክንያት የሲሚንቶ ዋጋ ጣራ ስለነካ ..የፊኒሺንግ.. ሥራ እንዲሰራላቸው የተዋዋላቸው ቤት አሠሪዎች ለቤቱ ሥራ የሚያስፈልገውን ሲሚንቶ አሁን ባለው ዋጋ ገዝተው ማቅረብ የማይችሉ መሆኑን ስለገለፁለት ነው፡፡ ቤት አሠሪዎች የሲሚንቶ ዋጋ እስኪቀንስና ለሥራው የሚሆን ሲሚንቶ እስኪገዙ ወይም በአማራጭ እጦት አሁን ባላው ዋጋ ገዝተው ሥራውን የሚያስጀምሩ ከሆነ፣ እጁን አጣጥፎ ከመቀመጥ ሌላ አማራጭ እንደሌለውም ሲነግረኝ፣ የሲሚንቶ ጉዳይ ምን ያህል አሣሣቢ እንደሆነ እንድረዳ አድርጎኛል፡፡ የሲሚንቶ

እጥረትና ዋጋ ንረት በየአቅጣጫው ምን ያህል ቤት እንደሚያንኳኳና ለስንቶች ሰቀቀን እንደሚሆን ሲታሰብ ጉዳዩ እንዲሁ በቀላሉ የሚታይ አለመሆኑንም ያመላክታል፡፡ ካለው የገበያ ፍላጎት አንጻር ሲሚንቶ ለማምረት ቅድመ ዝግጅት ላይ ያሉም ሆኑ ግንባታ ሥራ ውስጥ የገቡ የሲሚንቶ ፋብሪካዎች፣ ምርቶቻቸውን ለገበያ ለማቅረብ አመታት የሚጠብቁ መሆናቸውን ስንመለከት ደግሞ ነገሩ የበለጠ ያሳስባል፡፡ ለማንኛውም የሲሚንቶ እጥረትና የዋጋ ንረት ሊያስከትል የሚችለውን ሁለንተናዊ ጉዳት ሲጤን፣ አገሪቱ ለራስዋ የሚበቃ ሲሚንቶ እስከምታገኝ ድረስ አሁንም ብቸኛ የሆነው አማራጭ ምርቱን ከውጭ ማስገባት ነው፡፡

Precision/Recall Results

The performance of the summarizer is given below

- **Against ideal summary 1**: 0.71429
- **Against ideal summary 2**: 0.42857
- **Against ideal summary 3**: 0.28571
- **Average performance**: 0.47619

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all sources of materials for the thesis have been duly acknowledged.

_____

Eyob Delele Yirdaw

This thesis has been submitted for examination with my approval as an advisor.

_____

Dr. Dejene Ejigu

Addis Ababa, Ethiopia

March 2011