ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
Faculty of Informatics
Department of Information Science

Automatic Text Summarization for Amharic Legal Judgments

BY

HELENE ADANE

Name and Signature of Member of the Examining Board

Prof. B.R.K Rao, Chairman, Examining Board	
Dr Nega Alemayehu, Advisor	
Dr. Lars Asker, External Examiner	Lars Asker

Chairman, Faculty	Signature	Date
Chairman, Graduate Council	Signature	Date

AKNOWLEDGMENT

I would like to extend my gratitude to the almighty God with his mother St Mary for every thing done for me.

I would also like to thank my advisor Dr. Nega Alemayehu for his unreserved effort and tolerance to read and give constructive comments beginning from selection of title up until completion of the work.

My deepest appreciation goes to my beloved husband Semaw Nigatu for his kindness in every aspect of my life. All members of my family deserve gratitude for their caring and loving force towards me.

I am also indebted to Ato Sebsibe H/Mariam who helped me to write the source code using python and to Ato Belachew Chekene for giving me his laptop.

This thesis wouldn't have been realized without the support and encouragement of all the persons mentioned.

THANK YOU ALL

HELEN ADANE

DEDICATION

This Thesis is dedicated to my daughter Feven Semaw who was born while I was in the first year of this study.

TABLE OF CONTENTS

ABSTRACT

With the continuing fast growth of information and data today, it has become more and more urgent and important to find proper information efficiently, with some improved mechanisms.

Nowadays, people need much more information in work and life. The use of information technology such as the Internet makes information more easily accessible. However, people are having problems to easily get the information they want in a summarized way without wasting their time in the vast load of information made available to them. Thus, automatic text summarization draws substantial interest since it provides a solution to the information overload problem people face in this digital era.

This work is concentrated on producing a prototype system of text summarization on Amharic legal judgments. The methodology employed is an extraction technique.

Amharic legal judgments rendered by the supreme court of Ethiopia are selected in consultation with legal experts (lawyers, law instructor & students). The data selected in this manner are employed to generate the summaries.

Sentences in each judgment are classified in to different units according to their argumentative roles. From each argumentative unit, sentences with the highest weight at 20% compression rate are extracted and presented as a summary.

To evaluate the performance of the system, a random summary at similar compression rate is generated. Using extrinsic evaluation technique, the performance of the system summary and the random summary were compared with an ideal summary (human generated summary).

The results obtained from the system are promising when compared with the random summary.

To improve the performance of the system, sentences at 10% compression rate were extracted and the system's performance has improved. Therefore, the sentences extracted by the system summary using different extraction features are much closer to the manual (ideal) summary.

The prototype text summarizer has been developed using the Python programming language as a tool.

CHAPTER ONE

INTRODUCTION

1.1 Background

Legal judgments are made daily on different cases. In Ethiopia, courts are rendering legal judgments in different languages. Amharic legal judgments are one form of decisions given at federal and some regional courts.

Legal Judgments or decisions are rendered by courts of law to resolve disputes among individuals, associations, public enterprises etc. Accurate and timely decision-making requires relevant information to the case at hand. Judgments are given by courts on the basis of the facts presented by litigants, evidences, and analysis of legal provisions. Legal provisions (Laws) are the body of principles, standards and rules that the courts apply in the decision of controversies. The function of law is not limited to resolving disputes and molding the social behavior, it also serve as a means of maintaining peace and order and establishes organs of the government (corpus juris secundum vol. 49 pp.87).

The above-mentioned purposes of law can be attained only when there are legally established forums, which enter judgments by using the appropriate legal provisions.

However, legal provisions by themselves are not sufficient to arrive at decisions. To develop the reasoning capacity of judges, lawyers, and law students previously made decisions (precedents) play a significant role.

Precedents are used as a reference for or against a particular line of legal reasoning. Rule based decisions (precedents) are highly dependent on justifications or reasons provided by judges. To find solution to a legal problem not directly indicated in the law, lawyers look for precedents of similar cases.

For a single query in a database of law reports, we often receive hundreds of documents that are very long to study for which legal experts and law students request summaries. In order to make judgments accessible and to enable rapid scrutiny of their relevance, manual summaries are usually derived by legal expertise.

Manual summarization can be considered as a form of information selection using unconstrained vocabulary with no artificial linguistic limitations.

Decisions made every day need to be kept in accessible manner as a reference to decisions to be made in the future. However, since large numbers of decisions are made daily, referring to the whole document of a decision is time consuming and tiresome. Thus, it is indispensable to have summaries of the documents.

Summaries are texts themselves, which are intended to be taken as substitutions for the source documents. Text summarization is the process of

taking out the most important information from the whole text to produce an abridged version for a particular user or task (Lancaster, 1998).

With the coming of the information revolution, electronic documents are becoming a principal media of business and academic information. Thousands and thousands of electronic documents are produced and made available on the Internet each day.

In order to gain access and control the flood of information, every one needs to know in brief what is worth reading and what is useful for a particular purpose. Users prefer to read the essence of a document than all of it in order to save time. By giving an overview or outline of content, summaries save readers time and avoid unwanted details.

Having text summarization system would thus be immensely useful in serving these need

In most cases summaries are written by humans, but nowadays, the overwhelming quantity of information and the need to access the essential content of a document with shorter period of time creates a need for the formation and production of powerful computational tools in order to extract relevant information in a condensed form .This need makes automatic text summarization a major research field.

ATS (Automatic Text Summarization) is an active field of research in both IR (Information Retrieval) and NLP (Natural Language Processing) communities.

Summarization is important for IR since it is a means to provide access to large repositories of data in an efficient way. It shares some basic techniques with indexing, since both indexing and summarization are concerned with identifying the essence of a document. (http://www.summarization.com)

Summaries, serving as substitutes for the complete, full version of a document, are needed for many purposes. Some summaries actually inform the reader of the main contents of the document, others may simply indicate whether the complete text version should be of interest without explaining the text in detail. Summaries are also used as text surrogates in abstracting publications, and replace full texts in automatic text retrieval systems.

"The goal of automatic summarization is to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the users' or application's needs" (Mani ,2001).

"To summarize is to reduce in complexity, and hence in length, while retaining the essential qualities of the original" (Kupiec et al, 1995).

Generally, the need to automatic summarization of documents is increasing because:

- It saves the time needed to produce an abstract or a summary;
- It reduces the problem of shortage of skilled man power for manual summarization;

- It allows the user to revise quickly what they have already read; and
- It enables to produce a summary in a certain standard or consistent format.

The above-mentioned reasons make ATS a very important research issue that deals with the need to reduce a document in to smaller manageable size to see if the full information is relevant to our need's or not.

Automatic Text Summarization is a technique where an electronic document is entered in to a computer and a summarized text is returned, which is a non redundant extract from the original text .In automatic text summarization (ATS), the most relevant parts of a document are extracted and put together in a summary that is shorter than the original document (Dalianis, 2004).

Automatic text summarization can be used for:

- Summarizing newspaper text;
- Summarizing reports;
- Search in foreign languages and obtain an automatic summary of the machine translated text;
- Extracting key word and summaries of e-mail for SMS in mobile phones;
- Browsing summaries of a text for possible downloading from the internet to a WAP mobile phone; and

In most countries of the world documents pertaining to legal judgments (or decisions) are summarized automatically as this saves time and helps to maintain consistency.

The importance of summarization in the legal domain stems from the role that precedents play in common law. Already a number of content providers are providing access to manual summarization of legal judgments. An automatic system would enable immediate access to preliminary summaries and serve as an assisting technology in manual summarization. Automatic summaries might also be incorporated to provide dynamic, customized content in information retrieval systems. These kinds of systems have great utility both for learning and especially as a research aid for lawyers.

1.2 Statement of the problem and Justification of the study

Text summarization is an increasingly pressing practical problem. Documents on paper and electronic formats are available in growing numbers. Due to this, the user is confronted with vast quantities of text, usually beyond the capacity of not only to absorb it, but also even simply to search the material.

Texts are written in depth and users are compelled to see unwanted details. Summaries will help in reducing the amount of text a user must read in order to get what he needs.

Similarly, Amharic legal judgments suffer from the above-mentioned problems. Due to this, legal experts are forced to spend their precious time on reading a large volume of these documents and finding the relevant judgments for their cases. Mainly due to this problem, the average time required to decide on a case in Ethiopia is delayed significantly. And, justice delayed amounts to justice absent.

There are large volumes of legal judgments, which are very important if they are used by the experts and law students. In order to make these documents usable, the Federal Supreme Court and The Addis Ababa University, Faculty of Law under the auspices of the American Embassy have conducted a case-indexing project, to generate manual summary of selected judgments. These manually summarized reports were intended to save the time and resource of persons who are in the legal domain.

Even if huge amount of money was invested to accomplish the task, this manually summarized report is not able to attain the intended goal, as there are its own shortcomings. Since humans have individual differences, their differences would not enable them to maintain consistency .The other problem is the time that is spent in order to make this manual summary could be used to perform professional responsibilities. Therefore, it will be better, if a powerful computational tool to extract the relevant information from these judgments replaces this manual summarization.

To the best of my knowledge, there is no automatic Amharic text summarizer developed for legal judgments in Ethiopia. Therefore, it is considered necessary to conduct a research in this area and look into possibilities of developing an automatic summarization tool for Amharic legal judgment.

1.3 Objectives of the study

The general objective of this study is to develop a prototype for automatic Amharic text summarizer using the extraction technique for legal judgments.

On the basis of the general objective, the following specific objectives of the study are set:

- Review the existing text summarizers and algorithms developed for other languages such as English;
- Study and analyze Amharic legal text structures;

- Design and develop a prototype that will provide a model for automatic Amharic legal text summarizers; and
- Test and evaluate the model

1.4 Methods

In order to achieve the general and specific objectives the following methods are employed.

Literature related to automatic text summarization is reviewed. Since the study is on Amharic text summarizers, the nature of the language is investigated. In addition, the structure of Amharic legal judgments is studied. To accomplish this task, books, journal articles, and relevant websites are consulted.

1.4.1 Data Collection and Analysis

Sample Amharic documents are collected from judgments of the Federal Supreme Court, and these texts are preprocessed for the summary purpose. A total of ten judgments with their manual abstracts, which are generated by AAU law faculty students, are selected. In order to get the required information for the research and comments at the different stages of the experimentation and evaluation, discussion and interviews were conducted with domain experts. The technique that is employed for this study is the extraction technique. Extraction is much easier than abstraction because in this technique we pick up the most important sentences from the document and

return it to the reader, that means we do not need to rewrite the document by making linguistics analysis .To extract the important sentences in a text, sentences are weighted based on cue phrases they contain and the location of the sentences. Then, sentences with the highest weight are selected. The sentence weights are the sum of the weights of the component words and the weight assigned to them by their location.

1.4.2 Software selection

The Python programming language is used to develop the summarizer. Python is an extensible, interpreted, object-oriented programming language. It supports a wide range of applications, from simple text processing scripts to interactive Web browsers.

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming.

Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. http://www.python.org/

Python's Unicode string type stores characters from the Unicode character set. In this set, each distinct character has its own number, the code point. Unicode supports more than one million code points. Unicode characters don't have an encoding; each character is represented by its code. In Python code,

Unicode strings simply appear as sequences of characters, just like 8-bit strings. Text files always contain encoded data. Each character in the text is encoded as one or more bytes.

Python has two different string types: an 8-bit non-Unicode string type (str) and a 16-bit Unicode string type (Unicode).Reading and writing Unicode files from Python is simple.

1.4.3 Evaluation

In order to evaluate the performance of the system, we need to have two other forms of summaries: manual summary and random summary.

Manual summary as its name suggests is a summary done by experts. In this case law students generate the manual summary by extracting the most important sentences from the original documents.

Random summary means a summary where by sentences are randomly selected from the source document automatically with n number of sentences, where n is the number sentences selected for the summary by the system. The system summary refers to the summary made by this system. Comparison among these summaries depicts the performance of the system. Accordingly, the manual summary is compared with the system summary and the random summary. And the figures obtained are compared against to assess how effective the algorithm used actually is. These comparisons are calculated on the basis of precision (correctness), and recall (completeness),

Precision measures how much of the sentences extracted by the system are relevant. This means precision enable us to see the correctness of a summary.

Recall measures how well the system performs at extracting all the relevant sentences.

$$\text{Recall} = \frac{\text{number of correct responses by the system}}{\text{Total number of correct responses by expert}}$$

$$\text{Precision} = \frac{\text{number of correct responses by the system}}{\text{Total number of responses by the system}}$$

The average precision *Pavg* (system and random) and recall *Ravg* (system and random) are calculated by summing up individual results and normalizing (Radev et al, 2003).

That is:

$$Pavgs = \frac{\sum_{i=1}^{n} Psi}{n}$$

$$Pavgr = \frac{\sum_{i=1}^{n} \Pr i}{n}$$

$$Ravgs = \frac{\sum_{i=1}^{n} Rsi}{n}$$

$$Ravgr = \frac{\sum_{i=1}^{n} Rri}{n}$$

Where: P_{si} = precision of each system summary

P_{bi} = Precision of each random summary

R_{si} = Recall of each system summary

R_{bi} = Recall of each random summary

n = number of documents

1.5 Scope and Limitation of the study

This research is limited to single document summary. However, a document is considered as a collection of sub-documents. And the type of material that is going to be summarized in this study is written documentation or text. The materials analyzed and summarized will not include any tables, graphs or pictorial information. Due to the limited time available only ten judgments made in areas of the law of family and succession are used for the research.

1.6 Application of Results

The research will contribute for future researchers who may develop full-fledged automatic Amharic text summarizers. The results of this study can also be used as an input to develop text summarizer for other local languages.

This research will also help the judiciary, Justice and legal system research institutions and law schools to provide a brief summary of case reports.

1.7 Organization of the Thesis

This thesis is organized in five chapters. Chapter one is an introductory chapter. It acquaints readers of this paper with the background, problems that require solution, objectives to be achieved, the methodologies employed, and applications of the study.

Chapter two gives an overview of related research work and background introduction about Automatic Text Summarization. Chapter three constitutes two parts. The first part is on legal judgments, here the forms and contents of legal judgments are explained, and the importance of text summarization in the legal domain and related researches are discussed. The second part gives an idea about the Amharic writing system, its characters, punctuations, and some problems are highlighted. The fourth chapter contains the experimentation. Here, the system requirements, the data source and the data preprocessing are discussed. Then the output of the system and the evaluation results are analyzed.

In the final chapter conclusions are drawn and recommendations are forwarded to improve the systems performance for future researchers.

CHAPTER TWO

TEXT SUMMARIZATION: REVIEW OF LITERATURE

2.1 Introduction

In this chapter, the roots of text summarizations are discussed by considering some of the works done by different researchers in the area. The types of summaries and the techniques that are used by different researchers to develop these summaries and the evaluation methods that are currently available are provided in brief.

2.2 Roots of Text summarization

Early systems were developed in the late 1950's characterized by surface level approaches, for example exploiting thematic features (important pats) such as term frequencies. Luhn's (1958) research was based on the idea of counting the frequency of words. He initially assigned a value to each word and this value, within the context of a sentence, created a significance value for that sentence. The sentence with the highest significance value made up the summary.

Edmundson (1969) expanded his predecessor's (Luhn's) work in to an analysis of four different important parts: key words, cue phrases, location and

title. He did different experiments and find out that cue-title-location produced the best summary, and that the keyword feature was not significantly important.

Key method: it is similar to the word frequency criteria used by Luhn.Sentences are given a weight that is the sum of the weights of the component words.

Cue method: it uses a cue dictionary, which includes a list of words that receive a positive weight and a list of words with a negative weight. The significance value of a sentence is the sum of the weight of the component words

Title method: It is assumed that words occurring in titles and subheads are good indicators of contents. Sentences are given a significant value based on the number of title and subhead words they contain

Location Method: In this method weights are given to sentences on the bases of where they appear in a document. Sentences appearing in certain sections are assumed to be more indicatives of content than others.

Edmundson has discovered that the cue, title and location methods are more likely to agree on sentences to be selected than any combination methods involving the key procedure, leading Edmundson to conclude that the key procedure, based on frequency criteria only, was inferior to the other methods.

Rush, Salvador and Zamor (1971) developed summaries on the basis of cue phrases in the text that aimed at creating indicative abstracts, and were successful in the abstract sense, in so far as they were not simply extracts as modifications were made to the sentences.

In the 1990's, there was a renewed interest in text summarization. Kupiec et al (1995), introduced machine learning approach in to summarization. They tested each sentence for certain features, which produced a total probability of whether it would appear in the summary. The features they measured were: sentence length cutoff, fixed phrase paragraph, thematic words, and upper case words. The features were calculated using a Bayesian classifier.

Teufel & Moens (1997), use a learning method (to acquire the abstract-worthiness of a sentence) of a combination of five properties of that sentence. The five features are cue phrase, location, length of sentence, theme word identification, and title word. The additional techniques they used are:

Sentence length method: This is a method where all sentences under a certain length (threshold level) receive a zero score, and all sentences above the threshold a 1 score. This method is useful for filtering out captions, titles, and headings.

Thematic word method: This method tries to identify key words that are characteristic for the contents of the document. It concentrates on none stop word list words which occur frequently in the document, but rarely in the

overall collection. In theory sentences containing (clusters of) such thematic words should be characteristic for the document.

In recent years, text summarization has been enjoying a special attention. Most past efforts for single document summarization standard test sets and large-scale evaluations have been reported or made available by the TIPSTER SUMMAC Text summarization evaluation (Lin et al, 2002).

In the year 2001, the Document Understanding Conference (DUC) sponsored by the National Institute of Standards and Technology (NIST) started in the United States www.duc.nist.gov. The Text Summarization Challenge (TSC) task under the NTCIR (NIINACSIS test collection for IR system) project started in 2000 in Japan. Both aim to compile standard training and test collections that can be shared among researchers and to provide common and large-scale evaluations in single and multi document summarization for their participants. Recent developments in this area are not limited to the above workshops and conferences. There are also projects that are under taken by different research teams such as SewSum, FociSum, Summariest, TRESTLE, SUMMONS, and MultiGen.

2.3 Dimensions of summaries

It is well understood that people use documents for different purposes. It is likewise the case that there should be different types of summaries to support

this purpose. As such, it is necessary to examine what type of summarization would best suite for a specific task.

The following are set of parameters that can be employed to classify summaries.

- Coverage
- Selectivity
- Informativeness
- Recipient

2.3.1 Text coverage

Text coverage refers to the number of documents from which a short representation can be derived. Summaries can be made of individual texts (single document) or of collection of texts (multi document).

A single document summary is a summary made of individual texts by condensing it in to a shorter form. These individual texts can be books, speeches, presentations and so on. (Hutchnis, 1993)

On the other hand multi-document summary is a summary where two or more documents are evaluated at the same time, and summary is constructed from all of these documents. These types of summaries typically summarize a set of documents that are related in some fashion.

2.3.2 Selectivity

Summaries can attempt to cover all "important" aspects of text (global) or they can report only part of the contents (selective), typically selective summaries are made for specific purpose or clients.

2.3.3 Recipients

Summaries can be made for specific groups of recipients or readers (directed or query based summaries) that is they can be targeted to their particular needs. However, summaries can also be for general consumption (undirected or generic summaries), that is, for use in an information system where the background knowledge of users cannot be predicted. These summaries can be evaluated using task based evaluation techniques.

Query based summaries are user focused summaries, that is, they relay on a specification of a user information need, such as area of interest, topic, or query. This type of summary focuses on material of interest to the user. A user-focused summary will be a tailor made summary for a particular user, often in response to a query, which the user has.

Generic summary on the other hand provides the author's point of view. This type of summary captures the central ideas of a document and is designed to distill its salient points. It is a summary that treats all topics of a source document with equal weight.

2.3.4 Informativeness

On this parameter three types of summaries are identified: informative, indicative, and evaluative.

An informative summary reports on the factual data conveyed or the details of the opinions expressed. Informative summaries provide information on the salient aspects of a document, seeking to cover as many topics as possible. These summaries omit detail or supporting information and just cover the most important points of the document, summaries of this type are often used in place of the document as an overview, and are suitable for fulfilling a user's information need if they are browsing for information or have a general interest in the subject of the document. It reflects the content of the original text, possibly spelling out the arguments. This summary needs to include all the salient information that the source contains. And this summary can serve as a substitute for the full document.

Indicative summaries on the other hand, are meant to hit at the contents of the document. In the IR context, indicative summaries play an interesting role because they help the user in judging the relevance of the document, and in determining whether to consider full text retrieval. They assist a user who is searching for information and has a specific information need. This type of summary states only that certain topics were covered without conveying the precise content of the facts or opinions described in original text. They merely provide an indication of what the original text was about. The purpose of the

indicative summary is to be a reference tool, which should assist the reader or user as to whether they want to read the full text. It should include the key topics and give an indication an indication of the central topic of the original text or enough information to judge the relevance of the text.

An evaluative or critical summary involves making a judgment on the source text, such as a review. "A critical abstract evaluates the subject matter of the source, expressing the abstract's views on the quality of the work of the author." (Lancaster, 1997). This summary locates the content or opinions of the text within the context of other texts treating similar topics. Such summaries express the point of view of the author on a given topic.

In theory, all combinations are possible. In practice, some types are found more commonly together than others. Example: directed summaries are usually evaluative or selective; general summaries are normally indicative or informative. (Hutchnis, 1993)

2.4 Techniques to Summarization

There are two fundamental techniques to automatic text summarization that represent the end points of the continuum. Summarization delivers either an extract that is, verbatim rendition of some portion of a text or an Abstract that is, a compressed and reformulated version of the content. Both result in compression of text, but the extraction technique is relatively shallow, while the other is deep and complex.

2.4.1 The Extraction Technique

Summarization through text extraction is the creation of summaries using terms, phrases and sentences pulled directly from the source text using statistical analysis at a surface level. Occurrences of words or sentences are counted and analyzed according to their frequency and where they appear and reappear in the source text. This is sometimes referred to as "knowledge poor" processing and is rooted in the term weighting algorithms of information retrieval (Fen et al, 2004). This approach tends to represent information in terms of shallow features, which are then selectively combined together to yield a salience function used to extract information. These features are:

- Thematic features (key words) – presence of statistically salient terms based on term frequencies, this theory assumes that a term's salience is proportional to the frequency of the term in the document, but inversely proportional to the total documents in the corpus that contain the term;
- Location – position in text, position in paragraph, section depth, and particular sections;
- Background – presence of terms from the title or headings in the text;
- Cue words and phrases – picking up on certain phrases used in language can advise on the pertinence or redundancy of surrounding words and phrases. These can be domain specific 'bonus' and 'stigma' terms.

"An extract is a summary consisting entirely of material copied from the input" (Mani, 2001).

This method is very important since it is a very general technique, which will work without the system needing to be told beforehand what might be interesting or relevant information. But general methods for identifying abstract-worthy sentences are not very reliable when used in specific domains, and can easily result in important information being overlooked. Another limitation of this technique is only sentences or paragraphs in the original document are included, and often this is less semantic or syntactical analysis of the information than with an abstract.

An extracted summary remains closer to the original document, by using sentences from the text thus limiting the bias that might otherwise appear in a summary (McDonald &Chen, 2002).

Extraction methods are based on the hypothesis of the existence, in any text, of prominent textual units. Textual units are generally sentences, a set of sentences related by discursive links, or paragraphs. Here a selection algorithm is used, which is based on statistical or linguistic knowledge, or heuristic using distinct aspects, from which an ordered list of textual unit is obtained, according to some parameter (typically the percentage of the original text) given by the person who will read it. From this list a summary is built with textual units ordered as in the source text. Taking this in to account, a method for text summarization should be scalable.

Dalianis (2000) describes a set of summarization methods and algorithms based on extraction.

- Random: sentence order in text gives the importance of the sentences. First sentence highest ranking last sentence lowest ranking.
- Title: Words in title and following sentences gives high score.
- Term frequency (tf): Open class terms, which are frequent in the text, are more important than the less frequent.
- Position score: The assumption is that certain genres put important sentences in fixed positions. For example. News paper articles have most important terms in the four first paragraphs.
- Query Signature: The query of the user affect the summary in the way that the extract will contain these words.
- Average lexical connectivity: Number of terms shared with other sentences. The assumption is that a sentence that share more terms with other sentences is more important.
- Numerical data: Sentences containing numerical data obtain Boolean value 1 (is scored higher) than the ones without numerical values.
- Quotations: Sentences containing quotations might be important for certain questions from user.
- First sentence: First sentence of each paragraph are most important sentences.

Sentence Weighting

The first stage in summary generation is the weighting of the component sentences, as a way of generating a classification in importance of them, to obtain those more meaningful. Each criterion will assign a weight to text sentences. It must exist different weightings reflecting the distinct relevance it assigns to sentences.

There must also exist a hierarchy between the different criteria according the relevancy of each one.

According to Moncecchi and Prada (2000), the criteria used to assign weight to sentences are:

- Titles: this criterion assigns greater weight to text sentences containing words in the titles. To achieve it, it first identifies the words of each title. Once obtained, it emphases as meaningful those sentences containing appearances of such words. It is considered that a sentence containing all the words in the title will be more relevant than one that contains some of them, and this one, at the same time, more relevant than any text sentence not having them.
- Indicators and Indexes utilization: It uses certain linguistic units: indicators and indices. Here those sentences containing elements belonging to the set of indicators that determine the thematic announcement levels and their corresponding indices are considered.

Then sentences containing elements associated with the conclusion and summary levels, taking the sets that contain its respective indicators and indices are taken in to account.

- Section-border sentences: this criterion adds weights to all the sentences located at the beginning and end of sections. If the criterion is used to add new sentences to the summary, the previous weight will be 0 (previously selected once are not considered).
- Less-length sentences: this criterion is only used if one wants to select between two equal-weighted sentences at the moment of summary generation. It is based on the assumption that a sentence with less words, will probably give more information about the topic of the original text.

Despite the usefulness of sentence extraction methods in finding salient sentences, they cannot alone produce the highest quality extracts. Sentence selection techniques are often domain dependent. For example, the words "abstract" and "in conclusion" are more likely to appear in scientific literature than in newspaper articles. Position based methods are also domain dependent. The first sentence in the paragraph contains the topic sentence in some domains whereas it is the last sentence elsewhere. Combined with other techniques, however, these extraction techniques can still contribute to the quality of a summary (Goldstein J et al, 1999).

Watanabe, H (1996), on his paper describes a system, which automatically creates an abstract of a newspaper article by selecting important sentences of a given text. To determine the importance of a sentence several superficial features are considered and the weights for features are determined by multiple regression analysis of a hand processed corpus. The importance of a sentence is calculated as follows.

$$S = a + \sum wi * pi$$

Where: a is a constant

Pi is the number of points assigned to the i-th feature, which is normalized to be between 0 and 1

Wi is the weight assigned to the i-th feature.

The feature weight wi is calculated by multiple regression analysis of correct examples, which are abstracts created by testers.

The method Watanabe has proposed to calculate the importance of each sentence is using the sum of feature points multiplied by their feature weights. The most important sentences are then extracted as an abstract.

2.4.2 The Abstraction Technique

The other more complex and "knowledge rich" end point is summarization through abstracting. The aim here is to turn a computer generated analysis and synthesis of the source material in to a completely new, shorter text that is still cohesive and intelligible. This process is sometimes known as machine understanding, a multidisciplinary endeavor involving information retrieval, linguistics and artificial intelligence (Fen, 2004). In simplest terms automatic abstraction is fact extraction, compared to mere text extraction performed by statistical techniques. It is also called entity-level approach because it builds an internal representation for text, modeling text entities and their relationships. This approach tends to represent patterns of connectivity in the text to help determine what is salient. Relationships between entities include:

- Similarity – vocabulary overlap,
- Proximity – distance between text units,
- Thesaural relationships among words – by considering the semantic meaning of each verb or noun, we can detect relationships between different words,
- Semantic relations – based on parse trees.

Abstraction requires knowledge of the meaning of the information, and some ability to make inferences at semantic level. This technique needs intermediated techniques including passage extraction and linking, deep phrase selection and ordering, entity identification and relating, rhetorical

structure building and soon. According to Mani (2001), *"An abstract is a summary at least some of whose material is not present in the input".*

It considers summarization as a representation construction activity. They simulate the performance of a specialist over a given domain, which reads a text, understands it, and builds a summary about it. Within those approaches, a representation of the text built, which takes distinct form depending on the method. It can be a casual representation of the events in the text.

While the formation of an abstract may better fit the idea of a summary, its creation involves greater complexity and difficulty (McDonald & Chen, 2002).

2.5 Evaluation Methods for ATS

Techniques for automatic summaries have been a topic of discussion for as long as automatic summarization has been in existence, and are likely to continue for some time. The main reason for this is that there is no definition of an "ideal" summary, either an automatically generated summary one or one constructed manually by professional abstractor.

Jing et.al (1998), have analyzed the current and proposed evaluation techniques. According to them, Evaluation of summarization systems can be intrinsic or extrinsic. Intrinsic methods measure a systems performance in a particular task. Most evaluations of summarization systems to date are intrinsic: the qualities of the summaries are judged by direct human judgment of, for example, in formativeness, coverage, or fluency, or by comparing them

with an "ideal" summary. Recently a few task based evaluations were also presented. The performance of the summarization system was evaluated in an information retrieval or news analysis task.

2.5.1 "Ideal" Summary Based Evaluation

Most evaluations of summarization systems use an intrinsic method (Edmonson, 1969). The typical approach is to create an ideal summary either by professional abstractors or by merging summaries provided by multiple human subjects using methods such as the majority opinion union or intersection. The output of the summarizer is then compared with the ideal summary. Precision and recall are used to measure the quality of the summary. The main problem with this method is obvious- there is no single correct summary.

2.5.2 Task Based Evaluation

Jing et al (1998) stated that extrinsic methods evaluate the performance of a summarization system in a given task, such as GIMAT test, news analysis and information retrieval. TIPSTER III will also introduce task based evaluation of text summarization. Their evaluation will include approximately three tasks intended to judge the utility and appropriateness of the generated summaries, and to provide a way to measure improvement consistently.

Hahn (1997) as cited by Jing et.al (1998) describes more details of the proposed task based evaluation under TIPSTER. A "categorization task" will

be used to evaluate generic summarizers; systems will be scored on how well the summaries, in view of full text, can help users in categorizing documents in to different topics. An ad hoc information retrieval task will be used to evaluate user directed summarization systems. Time and accuracy will be used to measure system performance.

The requirements for evaluating the summarizer need to include a satisfactory random for comparisons. Since the manual extracts are available for each source document, the summary output can easily be statistically evaluated against these human generated summaries. However there is a need to compare all of the figures against some "outside" random because raw probability figures make it hard to assess how effective the algorithm used actually is. Therefore a random extract will be created for the specified compression rate.

CHAPTER THREE

LEGAL JUDGMENTS AND THE AMHARIC LANGUAGE

3.1 Introduction

This chapter has two parts; the first part is devoted to give a brief description about legal judgments. This includes the historical background in our country, their forms and contents. In the second part, the general structures of the Amharic texts are reviewed. The Amharic alphabets, punctuation marks, some problems faced when writing the Amharic texts and the Amharic computer fonts are treated.

3.2 Legal Judgments

Modern administration of justice dates back to 1925 E.C (Ethiopian Calander) when the Ethiopian Penal Code and other laws were enacted. Before that time judgments were given orally by customary courts in allover the country. Even at the time when litigations were believed to be relatively developed, legal judgments were rendered orally.

In 1934, the courts establishment proclamation number 2/1934 was promulgated. At this time different courts were established and after a year the then higher court under legal notice number 3/1935 enacted the procedural laws. At that time judgments were started to be made in written form. However, the judgments were written by reporters who simply record the decisions that are manually rendered by the judges.

Beginning from 1958, upon the enactment of the civil procedure code of Ethiopia, judgments started to be made in written by the judges themselves.

A judgment may be broadly defined as the decision or sentence of the law given by a court or other tribunal as the result of proceedings instituted therein, in this sense a decision of any court is a judgment.

A judgment is the judicial act of the court by which it accomplishes the purpose of its creation. It is a judicial declaration by which the issues are settled and rights and liabilities are fixed as to the matter submitted for decision. In other words a judgment is the end of the law; its rendition is the object for which jurisdiction is conferred and exercised. And it is the power by means of which a liability is enforced against the debtor's property.

A judicial judgment is not necessarily a judgment for money or thing enforceable by execution or other process; it may be a final and conclusive determination of a status, a right, a privilege, or the basis of action. A judgment is neither an action nor a special proceeding, but is the determination of an action or proceeding (Corpus juris secundum, vol 49).

A judgment constitutes the considered opinion of the court and is solemn record for expression and evidence of the actual decision of a law suit. The precedent or draft for judgment may not be treated as a judgment.

Legal judgments may be appealable or final. When Judgments are given by lower courts the party who is not satisfied with the judgments given by the court may bring appeal to the higher court when the judgment is not final. However, when decisions are final no appeal can be made from these judgments.

When appeal is made by the parties, the appellate court is required to summarize the judgments given by lower courts in addition to the pleadings made by the parties.

The Federal Supreme Court of Ethiopia, which was established in 1975 E.C, has an appellate jurisdiction over the decisions of the federal high court

rendered in its first instance jurisdiction and in its appellate jurisdiction in variation of the decisions of the federal first instance court.

3.3 Form and Contents of Judgment

The form of a judgment ordinarily is regulated by the practice of the court in which the judgment is rendered and under the procedure of some courts a duty rests on the successful litigant to see that the judgment is sufficient in form and substance.

To constitute a judgment there must be an express adjudication to that effect, but subject to the requirements of statute or court rule or practice, no particular form is required in a court proceeding to render its order a judgment, provided the rights of the parties may be ascertained there from.

Generally, the sufficiency of a judgment rests in its substance rather than its form, and although all informal judgment may be open to criticism, strict formality ordinarily is not essential to the validity of a judgment and if the record entry is sufficient in substance, mere irregularity or want of technical form will not render it invalid.

Normally, when judgment are prepared by judges they clearly state the name and address of the court, name of the judges presiding over the case, name of parties to a dispute, file number and title of the cases. Even if form is not a mandatory requirement in writing judgments the above stated elements are not only practiced but also very important.

A judgment is the expression of the opinion of a judge arrived at after due consideration of the evidence and of the arguments, if any, advanced before him. Judgments differ according to the nature of the trial, types of disputes, classes of courts and so many other factors.

Litigation begins with the drafting of the pleading and ends with the singing of the judgment. In drafting the pleading lawyer has to follow certain principles and also to take care to place his client's case before the court in as precise and concise manner as possible and in a form prescribed by law. A judgment has also to take no less a care in writing his judgment while deciding the case litigated before him.

Every litigation, civil or criminal, original or appellate, terminate in a judgment and it does certainly give resume of the entire litigation in the sense that the facts giving rise to the litigation the points for determination and the decision there on together with the reason therefore are almost always set out in the judgment. The essential requirements of a judgment under normal circumstances are:

1. Introduction: this part contains heading of the judgment including the title of the suit or proceedings.
2. Facts put forward by the plaintiff and the defendant in a civil case and the prosecution and the accused in and criminal case.
3. Points for determination (issues)
4. Decision on these points with reason thereof.
5. Final order granting or refusing to grant relief in a civil case and convicting the accused or acquitting him in a criminal case and awarding punishment in case of conviction.
6. Signature and designation of the judge and the date of decision.

A judgment is generally written in the language of the courts. The federal courts of Ethiopia are required to write their judgment in Amharic language where as regional courts make their judgment in their own regional language. The language should be plain and one which may be easily understood.

The title and heading of the judgment consist of the name of the court and that of the presiding officer, number of suit, trial or other proceeding and the names

of the parties. Every judgment is headed in the court of such and such judge or magistrate as the case may be. The number of the suit or proceeding and names of the parties follow thereafter.

The narration of the facts, the arrangement of the subject, including the discussion of the evidence and the flow of the language should be such that the cumulative effect may create an interest in one who read the judgment to go through it to the finish. The judgment must present a connected story of the entire case. The judgment is not intended merely for the appellate court, where it is scrutinizing on merits but also the parties and in many cases their descendant, generation after generation. The judgment must, therefore, be written in a form as may enable every one who cares to read it, and not merely those who are acquainted with the facts of the same, to follow and understand it.

In writing a judgment, it is difficult to give instruction as to how it should be written. The law can do no more than stating the essential requirements. Every judge has his own style of writing judgment. However, he is expected to keep these requirements of the law in mind while writing the judgment.

Essential Requirements of Judgments are:

1. Facts

In framing a judgment attention to its structure is of high importance. Then theme should be developed in logical sequence from the opening to the conclusion, so that the minds of the reader can follow the progress of the arguments with ease.

The normal course is first to let out the facts which have given rise to the question at issue. The narration of the facts in a judgment should be brief and concise.

to enable them to exercise, if they see fit and are so advised the right of appeal.

For the appellate court, this portion of the judgment assists to critically examine the decision of the lower court. The appellate court may disagree with the decision of the lower court judge for different reasons.

3.4 Text Summarization for Legal Documents

Legal Judgments are one form of reports. As ever larger amounts of legal documents become available electronically, interest to find these documents in a summarized form becomes increasing

The large volume of legal information in electronic form creates a need for the generation and production of powerful computational tools in order to extract relevant information in a condensed form.

Why previous legal decisions and their summaries? Court order generally gives a solution to a legal problem between two or several parties. The decision also contains the reasons which justify the solution and constitute a law jurisprudence precedent from which it is possible to extract a legal rule that can be applied to similar cases.

To find a solution to a legal problem not directly indicated in the law, lawyers look for presidents of similar cases. For a single query in a database of law reports we often receive hundreds of documents that are very long to study for

which legal experts and law students request summaries. Most summarization works has concentrated on the domain of news papers.

There are important differences between news style and the legal language: statistics of words, probability of selection of textual units, position of paragraphs and sentences, words of title and lexical chains relations between words of the title and the key ideas of the text, relations between sentences and paragraphs and structures of the text.

For judgments we can identify discursive structures for the different parts of the decision and assign some argumentative roles to them. Newspaper articles often repeat the most important message but, in law, important information may appear only once. The processing of a legal document requires detailed attention and it is not a strait forward to adapt the techniques developed for other types of documents to the legal domain.

3.5 Related research

The FLEXICON project (Smith and Deedman, 1987) generates a summary of legal cases by using information retrieval based on location heuristics, occurrence frequency of index terms and use of indicator phrases. A term extraction module that recognizes concepts, case citations, statute citations and fact phrases leads to a document profile. This project was developed for the decision reports of Canadian courts.

SALOMON (Moens et al, 1999) automatically extracts informative paragraphs of a text from Belgian cases. In this project a double methodology was used. First, the case category, the case structure and irrelevant units are identified based on a knowledge base representation as a text grammar. Consequently, general data and legal foundations concerning the essence of the case are extracted. Secondly, the system extracts informative text units of the alleged offences and of the opinion of the court based on the selective representative objects. SUM (Grover et al., 2003) examined the use of rhetorical and discourse structure in level of the sentence of legal cases for finding the main verbs. The methodology is based on (Teufel and Moens, 1997) where sentences are classified according to their argumentative role.

LetSum (Farzindar and Lapalme, 2004) system developed for producing short summaries for legal decisions based on the exploration of the document structures and thematic segmentation in order to produce table style summary for improving coherency and readability of the text.

In the department of information science at AAU, there are two works done in areas of Automatic Text Summarization both are in the domain of Amharic news. These are Kamil's paper on Amharic news summarizer using extraction technique and Teferi's work, which on the same domain but using machine learning approach. Unlike the above two papers, this work is concentrated on Amharic Legal judgment and in addition to this the extraction features used are different.

3.6 The Amharic writing system

Amharic is the official language of the Federal Government of Ethiopia. And is widely spoken through out different regions of the country. Amharic is a Semitic language of the Afro Asiatic language group that is related to Hebrew, Arabic and Syrian. The language uses its own alphabets, numbers, punctuation marks etc for its writing system (Saba, 2001).

As cited by Daniel (2003), the Blackwell Encyclopedia of writing systems defines the term writing system as "a set of visible or tactile signs used to represent units of a language in a systematic way".

The present Amharic writing system was adopted from the Ge'ez writing system, which was the language of literature in Ethiopia in earlier times. (Bender, et al, 1976).

3.6.1 The Amharic Characters

Amharic is a syllabic-alphabetic language which has thirty-three basic characters each having seven forms or orders for each consonant-vowel combination (Bender et al, 1976) cited by (Bizuneh, 2003).Each character occurs in one basic form and in six other forms which is made in accordance with the sound that goes with the symbol, the non basic forms are derived from the basic forms by following regular modifications.

For example the seven orders of the consonants ሀ ለ ቀ in the order is:

ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
ha	hu	hi	ha	he	h	ho
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
le	lu	li	la	le	l	lo
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
be	bu	bi	ba	be	b	bo

In the above examples, each symbol represents a consonant together with its vowel. The vowels are fused to the consonant form in the form of diacritic (accent) markings (Bethelhem, 2002).The diacritic markings are strokes attached to the base characters to change their orders. For instance the above ሀ (ha) is changed in to the second order ሁ (hu) by attaching "¬" and in to the third order ሂ (hi) by attaching the marking "L" and so on. In this way, the 33 core characters yield 231 distinct symbols.

Currently there is a debate whether the language is actually syllabic or alphabetic (Baye, 1997). Alphabetic writing systems are systems that present the consonants and the vowels separately such as the English and Greek language. On the other hand, syllabic writing systems are systems that combine both the consonant and the vowel together. Example: Amharic writing system.

However, (Baye, 1997) argues that Amharic is alphabetic on the grounds that each symbol can be broken down in to consonant and vowel phonemes which can be independently represented by separate symbols. He describes the Amharic script in terms of 27 consonants and 7 vowel phonemes.

In addition to the 231 basic characters, there are 4 labiovelars which have five orders. (Example: ቈ ቍ ቌ ቋ ቇ) and eighteen additional labialized consonants. Example :(ኍ ሟ ሯ ሷ ቧ ቯ ...). There is also character ኟ, which has also seven forms used to represent 'v'sound of words from other Latin based languages.

Except for zero the Amharic language uses Geez numbers; which are mostly used for writing dates.

3.6.2 Punctuation Marks

In Amharic there are 17 punctuation marks (Beletu, 1982), cited by (Bizuneh, 2003). In this section only some of the most commonly used punctuation

marks that are used both in hand written computer written text will be reviewed.

Two dots (: Hulet netib), similar to a space in other languages like English, are used to separate words. This punctuation mark is mostly used in hand written text but it is becoming a common practice to exclude it from computer written text, rather a space is being used as word separator. The equivalent full stop or period is four dots (:: Arat netib) and a comma is two dots with horizontal line on the top (፣ netela serez). An equivalent for semi-colon is two dots with horizontal line above and bellows them (፤ dirb serez). Apart from this, the language has borrowed some punctuation marks like (?, “, ’, /, \) from other languages.

3.6.3 Characteristics of the Amharic writing system

The Amharic writing system does not have distinction between capital and lower case letters, that means all letters have the same case any where in the text.

There are many features observed regarding the writing system of Amharic language, some of these may cause problems for computation.

3.6.3.1 Consonants with Different Forms

Amharic borrowed most of its scripts from Geez. However, it did not select from Geez alphabet those symbols that are only necessary for its consonants. As a result there are certain phonemes with different symbols, where they have meaning in Geez but their meaning is not known in Amharic.

Example:	consonant		other forms
	ሀ	(ha)	ሐ ኀ
	ሠ	(sa)	ሰ
	አ	(a)	ዐ
	ጸ	(tsa)	ፀ

The pronunciation of these paired characterizes the same and each of them has their own orders.

The distinction between these symbols in Geez is when spelling certain words, however there is no rule as for their usage in Amharic language and as (Getachew, 1987) stated, the proper usage of these symbols is not studied exhaustively and also there is no standard dictionary to refer to. Therefore, it is not clear whether one should write" haymanot" (religion) as"ሀይማኖት፡, ሐይማኖት፡ or ኀይማኖት፡", also (Beletu, 1982) noted the confusion regarding the first order and the fourth order of some consonants. For example it is not clear which to choose (ሀ as ሀይሉ፡ or ሃ as ሃይሉ፡) to write 'Hailu' name of person. Due to this there arises some confusion and inconsistencies in Amharic writing, because the rules are known only by language experts or persons who know the Geez language.

3.6.3.2 Different forms of writing compound words

There exists different ways of writing compound word without affecting their meaning (Bizuneh, 2003), that means at one time the compound noun can be written as two separate words and at another time as a single word. For example: - ማዕድ ቤት፡ and ማዕድቤት፡ (kitchen).

3.6.3.3 Different ways of writing the same word

This is a problem that exists when the language has some words having different forms of writing system. For example:- the word " wesnoal" (he has decided) can be spelled as " ወስኖአል", "ወስኗል" or "ወስንዋል" which are different variants of the same word.

3.6.3.4 Problems regarding Abbreviations

In Amharic, it is also found that there is no consistency while writing abbreviations. For example "ዓመተ ምህረት"can be abbreviated as"ዓ.ም " ,"ዓም " and"ዓ\ም " ,similarly the use of hyphen is also not consistent: the same word "ዓመተ ምህረት " can also be written as "ዓመተ-ምህረት " .Therefore ,these kinds of words should come in to the a common word.

3.6.4 Computer Fonts

There are several Amharic fonts that have been developed; some of these include Agafari, Wasra, Visual Geez, Power Geez, Alpas, etc... (Saba, 2001).

All these Amharic software fonts have succeeded in helping users to enter and edit Amharic text with computers (especially for word processing purpose). However, all face some problems especially when they are used for database purpose. (Ethiopia, 2002).

CHAPTER FOUR

EXPERIMENTATION

4.1 Introduction

This chapter is devoted to describe the system which includes the data source; the data preprocessing that is manually done. The system's architecture, which is the steppingstone for the experimentation is also included here.

Based on the adopted techniques an algorithm of extraction is developed and using python programming language the algorithm is coded for practical application of the extraction.

Basic functioning principle of the system, the conducted tests with the results obtained and the evaluation of the system are included in this chapter.

4.2 Description of the System

The aim of text summarization is to identify the most important information in a document or set of related documents and conveying it in less space than the original text. This means that it gives an accurate and complete idea of the source document and save the reader's time. In other words, it can be said that text summarization is taking document source as an input and extract content from it, then give the most important information to the user as an output in a condensed form and in a manner sensitive to the user's need.

In line with this, the objective of this research is to develop a prototype system that can take Amharic legal text as an input and give summary as an output.

The system uses extraction technique, which selects the most important sentences from each thematic unit and presents it as a summary. The summary will be indicative and generic. The extraction is on the basis of the

sentence weights given to each sentence based on two features: location and cue phrases.

The system requirements are based on establishing a prototype text summarizer, and the emphasis is more on the production of a summarization algorithm to produce an automatically generated summary that are comparable to human generated summaries, and with a satisfactory evaluation method. This system is not required to have a user interface with a lot of functionalities.

4.3. The Data Source

The data collection contains 10 judgments of the Supreme Court of Ethiopia, which is the highest judicial organ. The judgments are selected from the book entitled "selected judgments of the Supreme Court of Ethiopia". From 1998 to 2001 a case-indexing project was made by the Federal Supreme Court in cooperation with the Addis Ababa University, Faculty of Law under the auspices of the American Embassy. Selected judgments of the Supreme Court of Ethiopia were compiled at that time.

A total of 10 judgments are collected from this document for two main reasons:

- The compiled judgments are deemed to be best judgments of the country; and
- Since the judgments are already collected from different court of law it saves time, energy and money.

All judgments are written in Amharic. After some preliminary considerations (i.e., discussion with legal experts), from different areas of law, in this research decisions made on the basis of the law of family and succession are collected. The law of family governs the relationship that exists among members of a family, while the law of succession regulates issues that may arise in relation to the rights and duties of the deceased person. Discussions made with judges

and other legal experts revealed that these types of cases are very important and increase in number rapidly. Therefore, developing a text summarizer for this kind of judgment is pivotal.

4.4 Data Pre-Processing

The first preprocessing that is made to the selected documents is to convert them from printed format to electronic format. The electronic format was written by using the visual Geez Unicode font.

As stated in chapter three, a legal judgment is composed of different units. At this stage the different units of these judgments are sorted out. This is done manually in consultation with legal experts in the field. Merging textual units dealing with the same subject in one theme makes this thematic segmentation. In this case these Amharic legal judgments have five categories of textual units. For thematic segmentation the following information are used: the presence of significant section titles, the position of a segment, and certain linguistic markers.

Thus, the textual units of a typical judgment that is written in Amharic are:

- Introduction: contains the section name of the jurisdiction, the place of the hearing, names of parties, name of the judge and document file number. It describes the situation before the court and answers the questions: who did what? To whom?
- Background: it describes the cause of action, decisions of the lower court and reasons for appeal.
- Fact: - the sentences recount the events or circumstances which gave rise to legal proceedings in depth.
- Judicial Analysis: describes the comments of the judge and finding of facts, and the application of the law to the facts as ground. For the legal expert, this section of judgment is very important because it gives a

solution to the problem of the parties and leads the judgment to a conclusion. Here, the judge frames or identifies the issue that needs ruling and give his/her well thought reasoning on the bases of the law and evidences produced.

- Disposal: A sentence which either credits or discredits a claim or previous ruling. This is the nucleus of the judgment. On the basis of the analysis given, the judges enter in to judgment.

4.5 Architecture of the System

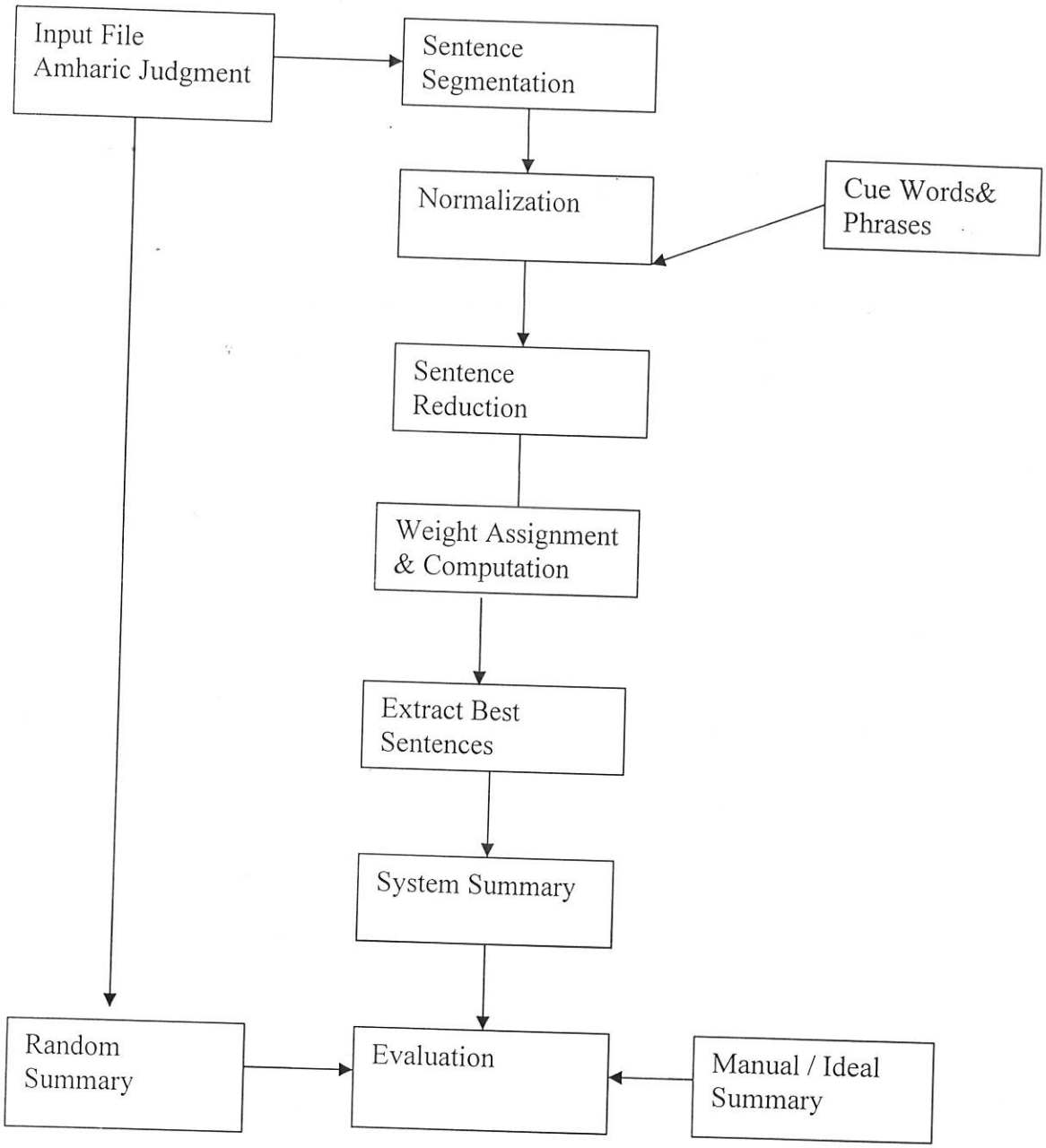

Figure 4.1. The system flowchart

Sentence segmentation: identify each sentence and headline from the original documents and removes the key word of the headlines.

Normalization: it replaces some variants of characters by a single one from both the original document and the cue phrase file. These characters are characters that are pronounced in the same fashion. These are:

Ha = ሀ, ሃ, ሐ, ሓ, ኀ, ኃ

Hu = ሁ, ሑ, ኁ

Hi = ሂ, ሒ, ኂ

Hie = ሄ, ሔ, ኄ

H = ህ, ሕ, ኅ

Ho = ሆ, ሖ, ኆ, ኍ

Se = ሠ, ሰ, ሡ

Su = ሡ, ሢ

Si = ሢ, ሢ

Sa = ሣ, ሣ

Sie = ሤ, ሤ

S = ስ, ሥ

So = ሦ, ሦ

A = አ, ዐ, ኣ, ዓ

U = ኡ, ዑ

I = ኢ, ዒ

Ie = ኤ, ዔ

I = እ, ዕ

O = ኦ, ዖ

TSe = ጸ, ፀ

TSu = ጹ, ፁ

TSi = ጺ, ፂ

TSa = ጻ, ፃ

TSie = ጼ, ፄ

TS = ጽ, ፅ

TSo = ጾ, ፆ

SHe = ሸ, ሼ

GNe = ኘ, ኜ

Ce = ፈ, ፌ

Che = ቸ, ቼ

Wu = ዉ, ው

Je = ጀ, ጄ

All these Amharic letter characters have the same pronunciation. Thus if the system finds any of these characters it replaces all by the first letters.

Normalization is not encouraged by the language experts specially those who have knowledge of the mother language "Geez "since some words may change their meanings due to change in the characters. However, most people when writing in Amharic they don't know which letter is appropriate to express their ideas. Therefore, to avoid this problem normalization is used.

Sentence Reduction: to reduce the length of the sentences without losing their original idea, quotations and dates are removed. The quotations are proclamation numbers and legal provisions (article numbers).

Example: removal of article numbers like

በመዝገብ ቁጥር 276/3

የፍትሐ ብሔር ሕግ ቁጥር 274(2)

የፍ/ሕ/ቁ/ 543

ሐምሌ 20 ቀን

Weight computation: at this point the system will assign weight to a sentence based on two features: cue phrases and sentence location. The system first reads the normalized cue phrase file and identifies maximum cue phrase length. After that the system reads the input file and checks each word and phrase in the cue phrase file. If the word or phrase is in conformity with the list, the system assigns value "1" to the word or the phrase. However, when there is mismatch, it assigns value zero (0). After reading a single sentence, the system assigns value to the sentence on the basis of its location. Finally, it computes the total weight of the sentence.

Sentence weight *(Sw)* = $$\frac{\sum_{i=1}^{n} swci + swl}{N+1-J}$$

Where *Swci:* The weight assigned to each component word of a sentence

Swl: The sentence weight assigned using sentence location. *Swl*=1; if the sentence is located at the beginning or at the end of a segment. 0; if the sentence location is other than beginning or end of a segment.

N+1: The total number of words in the sentence plus 1 for the weight based on location (here 1 is added to normalize the sentence weight: that is if it is not added sentences that have value 1based on their location will always be selected).

J: is the number of extra words where:

$$J = \sum_{i=1}^{n} wpi - n$$

Where: *wpi:* is the number of words in the phrases

n: is the number of phrases in a sentence

The above formula, which is used to calculate the sentence weight, has similarity with the formula that Watanabe has used. The difference lies on the features used to extract the sentences and on the weight calculation. Instead of using multiple regression analysis to assign weights of features in this paper the weights of features are assigned according to the writer's intuition, that is, a value 1 is assigned for important features and 0 for others. Thus the formula used by Watanabe is modified so as to fit with the domain and the extraction features used in this paper.

Extract best sentence: after calculating the weights of each sentence, the system will extract **n** best sentences from each segment using the compression rate that is given. The sentences that are related are sentences with the highest weights. At this point, if two sentences have the same weight, the best sentence to be selected is the shortest one, which is a sentence with lesser number of component words.

System summary: In this part all the sentences that are extracted from each segment are put together in an output file as a summary to the whole document.

4.6 EVALUATION

The aim of the evaluation program is to evaluate or to measure the effectiveness of the system program at creating summaries. To do this, the program needs measures to rate itself against. Therefore, the evaluator program will need to introduce the extract to be evaluated (system summary), a human generated summary (supplied as part of the document), and automatically generated random summary (random summary)

The two automatically generated summaries, that is, the system summary and random summary are evaluated against the human generated extract to produce the overlap of how many sentences appear in both the automatic summaries and the human generated one. This is done using precision and recall.

Algorithm of the system

The following algorithm is developed in order to build the prototype system text summarizer

Take the name and location of input file, cue phrase, and the system and random output files.

Try to open the files.

If it fails to open, the system will write "unable to open the file"

Normalize:

Replace variants of characters that have similar pronunciation with one of these characters.

Escape the lines until the system finds the first key word.

Escape the line which contains the key word and read the next line until it reaches end of a sentence that is (::)

Check for cue words or phrases in the sentence.

If there is any, assign weight 1 else assign weight 0.

Then assign weight to the sentence based on the location features.

If the sentence is located at the beginning or at the end, assign a value 1 otherwise give 0.

On the basis of the formula given calculate the weight of the sentence.

Repeat steps 5-8 for all sentences until the next key word is fond.

Select sentence/s with the highest weight using the compression rate that is given.

When the next key word is found repeat steps 5-10.

4.7 Testing and Test Results

The system is tested to ensure that it is producing the intended results. The ten collected judgments are tested to guarantee that the feature values are accurately allocated and totaled up, and the summary is made at the specified length given .A sample of these documents were assessed manually. Fore Example: The following Amharic legal judgment is one of the samples that are tested.

መግቢያ
ጠቅላይ የንጉስ ፍርድ ቤት
የፍሐብሔር ችሎት
ይግባኝ ባይ፡- ኮሎኔል ማንደፍሮ አመኑ
መልስ ሰጭ፡- ተዋበች ተክለ ማርያም
የይመ.ቁ.371/60
ዳኞች፡- አፈንጉስ ተሾመ ኃይለማርያም፣ ታደሰ ተክለ ጊዮርጊስ፣ ኃይሌ በፀሎት፡፡

ምክንያት

ይህ ይግባኝ በይጣራ ከዙፋን ችሎት ታዞ የቀረበ ነው፡፡ መዝገቡን መርምረናል፡፡ መልስ ሰጭ ክስ የመሠረተችው ይግባኝ ባይ ሚስት አድርጌ አግብቸሻለሁ ብሎ ወስዶኝ እንደ ባልና ሚስት ሆነን ስንኖር ልጅ ስለወለድኩለት ከመድኃኒት፤ከሐኪም፣ ለፅዳት፣ ለልብስ ያወጣሁትን፣ አብረን በነበርን ጊዜ ስለደመወዝ የድካም ዋጋዬን እንዲሁም ወደፊት ለልጅቷ ማሳደጊያ የሚሆን ይከፈለኛል በማለት ነው፡፡ ይግባኝ ባዩ በበኩሉ መልስ ሰጭ በክሷ የዘረዘረችውን ክዶና ተቃውሞ ተከራክሯል፡፡ ክሱ የቀረበለት የአውራጃው ፍርድ
ቤት ይግባኝ በዩ ክስ የቀረበበትን ገንዘብና እንዲሁም ለልጅቱ ማሳደጊያ ይከፍላል በማለት ፈርዷል፡፡ ከፍተኛው ፍርድ ቤትም ነገሩን በይግባኝ ካየው በኋላ ለሃኪምና አስፈላጊው ጉዳይ ወጭ ለሆነው ብር 500 ብር ብቻ በማለት አሻሽሎ በሌላው እረገድ የአውራጃው
ፍርድ ቤት የፈረደውን አጽንቶ ይግባኙን አሰናብቷል፡፡ የዚህ ፍርድ ቤት ይግባኝ በፍርድ ምርመራ በኩል ከታዬ በኋላ በባልና
ሚስትነት መኃባታቸውን የሚያስረዳ አንዳች ነገር አልተገኘም፡፡ ከተከሳሹ ለመውለዷ ተከሳሹ ለስራ በግርድና ቀጥሯት እንጂ አንኳን
በሚስትነት ላገባት የግብረ ሥጋ ግንኙነት የለኝም ብሎ ምሎ መስክሯል፡፡ ሌሎችም ምስክሮች እንደጭብጡ አልመሰከሩላትም፣

ባልተጣራ አኳኋን የተፈረደ መስሎ ስለሚታይ ጠቅላ የንጉሥ ነገሥት ፍርድ ቤት እንዲያጣራው ቢያደረግ ይሻላል በማለት
ሐሳብ አቅርቦ በዚሁ መሠረት እንዲጣራ ስለታዘዘ ሁለቱንም ወገን አቅርበን አከራክረናል፡፡

ፍሬ ነገር

የይግባኝ ባዩ ክርክር በፍሬ ነገርና በሕግ ላይ የተመሠረተ ነው፡፡ በፍሬ ነገር በኩል ያለው ክርክር ገረድ እንድትሆነኝ ከመቅጠሬ
በቀር በሚስትነት አላገባኋትም የሚል ሲሆን በሕግ በኩል የቀረበው ክርክር ደግሞ በፍታብሔር ሕግ ቁጥር ከ 629፣ እስከ 638፣
707፣ ከ718፣ እስከ 721፣ ከ646፣ከ 758ነ እስከ 761፣ በተግፉት ድንጋጌዎች ላይ የጠመሰረተ ነው፡፡

ጭብጥ

መልስ ሰጭ በአገር ልማድ ወይም በሕጉ መሠረት የጋብቻ ውል ተፈፅሞ የተገባት ሚስት አይደለችም፡፡ ቢሆንም ይግባኝ ባይ
እንደሚለው በግርድና የተቀጠረች ገረድ ናት ለማለት አልተቻለም፡፡ በምስክሮች ቃል እንደተነገረውና ራሱ ይግባኝ ባዩ የጻፈው
ደብዳቤ እደሚያስረዳው መልስ ሰጭ ከይግባኝ ባዩ ቤት የገባችው በግርድና ሳይሆን ሚስት አደርግሻለሁ ስላላት መሆኑን ለመገንዘብ
ችለናል፡፡ ምንም እንኩዋን ሕጋዊ የጋብቻ ውል ሳትፈፅም ከይግባኝ ባዩ ቤት ገብታ በመገኘቷ ሕጋዊ ሚስት ናት ለማለት ባይቻልም
አንባቢው ሁኔታና ማስረጃው ሲታይ፣ ሕጋዊ ጋብቻ ሳይፈጽሙ እንደባልና ሚስት ሆነው ይኖሩ ነበር ለማለት ይቻላል፡፡

ምርመራ

ሕጋዊ ጋብቻ ባይኖርም አንድ ወንድና ሴት እንደባልና ሚስት ሆነው በሚኖሩበት ጊዜ የተጸነሰው ወይም የተወለደው ልጅ
የዚሁ ከሴትዮዋ ጋር እንደባል ሆኖ የሚኖረው ሰው ልጅ ነው ተብሎ እንደሚገመት በቁጥር 745 የተጻፈው ይደነግጋል፡፡ በዚህ ሕጋዊ
ግምት ላይ ማናቸውም መቃወሚያ ማቅረብ የማይቻል መሆኑን በቁጥር 743፣ ንዑስ ቁጥር (2) የተጻፈው ድንጋጌ ይናገራል፡፡

ውሳኔ

ልጅቱ ይግባኝ ባይና መልስ ሰጭ እንደባልና ሚስት ሆነው ሲሆኑ የተፀነሰች ለመሆኗ መልስ ሰጭ አስረድታለች፡፡ ይግባኝ ባይ ግን ይህን በመቃወም ያቀረበው ማስረጃ የለም፡፡ ስለዚህ በሕጉ መሠረት የይግባኝ ባዩ ልጅ ናት ብለን የከፍተኛው ፍርድ ቤት የፈረደውን አፅንተናል፡፡ ለከፍተኛው ፍርድ ቤት ይጻፍ፡፡ ይህ ፍርድ ዛሬ ሐምሌ 4 ቀን 1961 ዓ.ም አዲስ አበባ ተሰጠ፡፡

The summary obtained from the system contains 6 sentences, which is approximately 20% of the original document. In addition a random summary is generated to serve as a comparison in order to measure the performance of the system. These two summaries are shown bellow.

System summary based on sentence extraction features:

ይህ ይግባኝ በይጣራ ከዙፋን ችሎት ታዞ የቀረበ ነዉ::
ይግባኝ ባዩ በበኩሉ መልስ ሰጭ በክሷ የዘረዘረችዉን ክዶና ተቃዉሞ
ተከራክሯል:: ክሱ የቀረበለት የአዉራጃዉ ፍርድ ቤት ይግባኝ በዩ ክስ የቀረበበትን
ገንዘብና እንዲሁም ለልጅቱ ማሳደጊያ ይከፍላል በማለት ፈርዷል:: የይግባኝ ባዩ
ክርክር በፍሬ ነገርና በህግ ላይ የተመሰረተ ነዉ::
መልስ ሰጭ በአገር ልማድ ወይም በህጉ መሰረት የጋብቻ ዉል ተፈጽሞ የተገባች
ሚስት አይደለችም::
ህጋዊ ጋብቻ ባይኖርም አንድ ወንድና ሴት እንደባልና ሚስት ሆነዉ በሚኖሩበት
ጊዜ የተጸነሰዉ ወይም የተወለደዉ ልጅ የዚሁ ከሴትዮዋ ጋር እንደባል ሆኖ
የሚኖረዉ ሰዉ ልጅ ነዉ ተብሎ እንደሚገመት በቁጥር 745 የተጻፈው ይደነግጋል::
ልጅቱ ይግባኝ ባይና መልስ ሰጭ እንደባልና ሚስት ሆነዉ ሲሆሩ የተጸነሰች
ለመሆኗ መልስ ሰጭ አስረድታለች::

Random summary that is randomly generated without taking in to consideration any extraction features:

መዝገቡን መርምረናል:: መልስ ሰጭ ክስ የመሰረተችዉ ይግባኝ ባይ ሚስት አድጌ
አግብቸሻለሁ ብሎ ወስዶኝ እንደ ባልና ሚስት ሆነን ስንኖር ልጅ ስለወለድኩለት
ከመድሀኒት፤ከህክምና፣ ለጽዳት፣ ለልብስ ያወጣሁትን፣ አብረን በነበርን ጊዜ
ስለደመወዝ የድካም ዋጋዬን እንዲሁም ወደፊት ለልጅቷ ማሳደጊያ የሚሆን
ይከፈለኛል በማለት ነዉ:: ከፍተኛዉ ፍርድ ቤትም ነገሩን በይግባኝ ካየዉ በኋላ
ለህክምና አስፈላጊዉ ጉዳይ ወጭ ለሆነዉ ብር 500 ብር ብቻ በማለት አሻሽሎ
በሌላዉ እረገድ የአዉራጃዉ ፍርድ ቤት የፈረደዉን አጽንቶ ይግባኙን አሰናብቷል::
ከተከሳሽ ለመዉለድ ተከሳሹ ለስራ በግርድና ቀጥሯት እንጂ አንኳን በሚስትነት
ላገባት የግብረ ስጋ ግንኙነት የለኝም ብሎ ምሎ መስክሯል:: በፍሬ ነገር በኩል
ያለዉ ክርክር ገረድ እንድትሆነኝ ከመቅጠሬ በቀር በሚስትነት አላገባኋትም የሚል
ሲሆን በህግ በኩል የቀረበዉ ክርክር ደግሞ በተጻፉት ድንጋጌዎች ላይ የጠመሰረተ
ነዉ:: መልስ ሰጭ በአገር ልማድ ወይም በህጉ መሰረት የጋብቻ ዉል ተፈጽሞ
የተገባት ሚስት አይደለችም::

These two summaries were compared with a human generated summary to identify which one is closer to the manual summary. In this case the manual summary is taken as the ideal summary, which is a summary that represents the source document.

The human generated summary:

መልስ ሰጭ ክስ የመሠረተችው ይግባኝ ባይ ሚስት አድርጌ አግብቸኛለሁ ብሎ ወስዶኝ እንደ ባልና ሚስት ሆነን ስንኖር ልጅ ስለወለድኩለት ከመድኃኒት፤ከሐኪም፣ ለፅዳት፣ ለልብስ ያወጣሁትን፣ አብረን በነበርን ጊዜ ስለደመወዝ የድካም ዋጋዬን እንዲሁም ወደፊት ለልጅቷ ማሳደጊያ የሚሆን ይከፈለኛል በማለት ነው። የይግባኝ ባዩ ክርክር በፍሬ ነገርና በሕግ ላይ የተመሠረተ ነው። መልስ ሰጭ በአገር ልማድ ወይም በሕጉ መሠረት የጋብቻ ውል ተፈፅሞ የተገባት ሚስት አይደለችም።ሕጋዊ ጋብቻ ባይኖርም አንድ ወንድና ሴት እንደባልና ሚስት ሆነው በሚኖሩበት ጊዜ የተጸነሰው ወይም የተወለደው ልጅየዚሁ ከሴትዮዋ ጋር እንደባል ሆኖ የሚኖረው ሰው ልጅ ነው ተብሎ እንደሚገመት። ስለዚህ በሕጉ መሠረት የይግባኝ ባዩ ልጅ ናት ብለን የከፍተኛው ፍርድ ቤት የፈረደውን አፅንተናል።

The evaluator, which is a separate module, calculates the precision and recall of both the system summary and the random summary against the human summary. The table bellow shows the results.

$$\text{Precision} = \frac{\text{Number of relevant sentences extracted}}{\text{Total number of sentences extracted}}$$

$$\text{Recall} = \frac{\text{Number of relevant sentences extracted}}{\text{Total number of relevant sentences}}$$

Document number	Precision		Recall	
	System	Random	System	Random
1	0.55621	0.14043	0.88679	0.31132
2	0.3494	0.22093	0.57426	0.56436
3	0.33858	0.35714	0.57333	0.73333
4	0.52632	0.36905	0.54945	0.34066
5	0.36082	0.11735	0.7	0.23
6	0.21839	0.18301	0.49565	0.48696
7	0.29245	0.29126	0.45588	0.44118
8	0.26471	0.13492	0.61017	0.28814
9	0.27642	0.34167	0.44156	0.53247
10	0.20958	0.21702	0.47297	0.68919
Average	0.33929	0.23728	0.57601	0.46176

Table 4.1 Precision Recall table at 20% compression rate

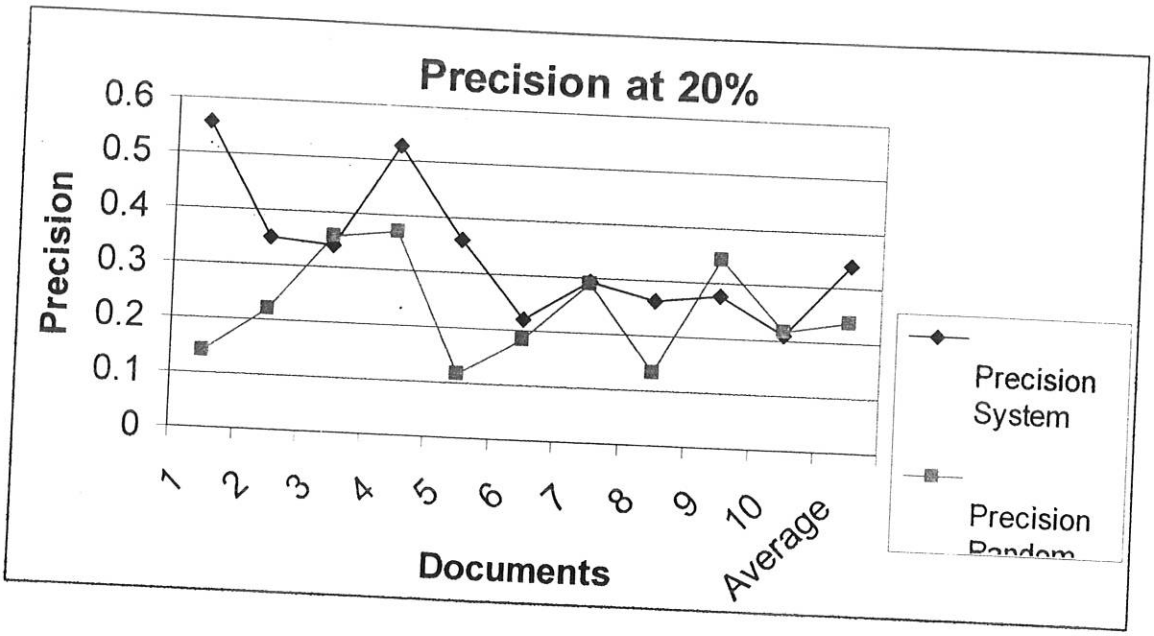

Figure 4.2 Precision graphs of system and Random summary at 20% compression rate

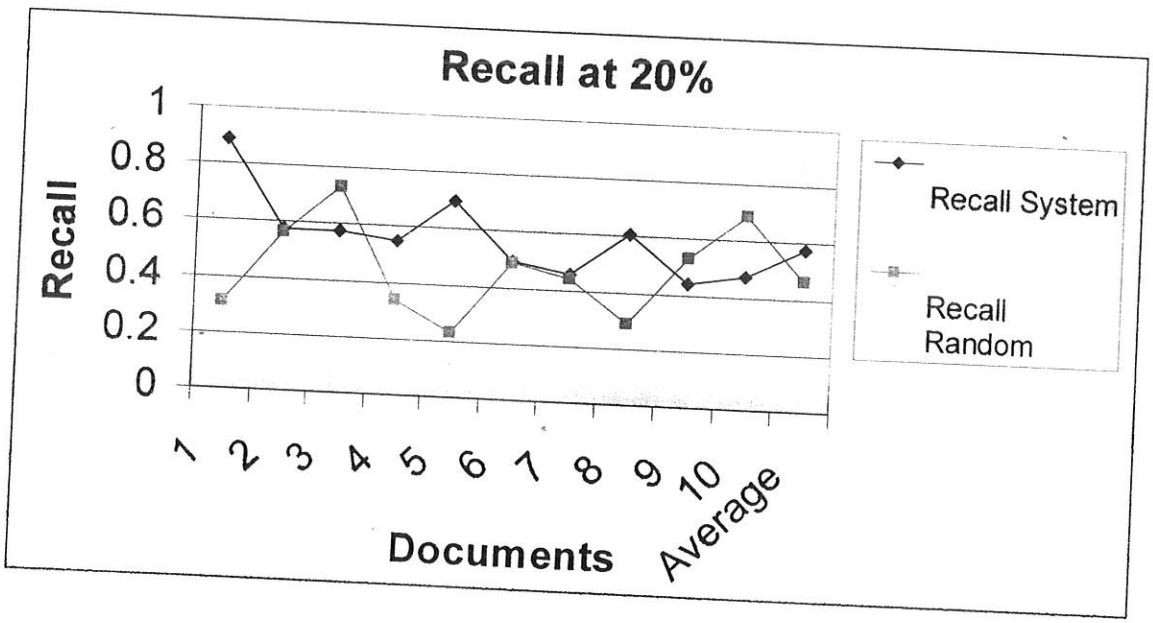

Figure 4.3 recall graph of system and Random summary at 20% compression rate.

The above table and graphs show that the system summary is nearest to the manual summary and its recall is also better than the random summary.

Precision of the system summary =33.9%

Precision of random summary =23%

Recall of system summary =57%

Recall of random summary =46%

In an attempt to improve the performance of the system and to reduce volume of the summary the compression rate is scaled down from 20% to 10%. It is also tested against the human generated summary. And the performance result is shown in the table bellow.

Performance at 10% compression rate

Document number	Precision		Recall	
	System	Random	System	Random
1	0.68468	0.34454	0.71698	0.38679
2	0.45763	0.19737	0.53465	0.29703
3	0.4375	0.35417	0.56	0.45333
4	0.65714	0.4661	0.50549	0.6044
5	0.41176	0.24603	0.7	0.31
6	0.17544	0.16364	0.34783	0.23478
7	0.33766	0.45588	0.38235	0.45588
8	0.22222	0.125	0.40678	0.11864
9	0.28571	0.21374	0.44156	0.36364
10	0.24638	0.20889	0.45946	0.63514
Average	0.39161	0.27753	0.50551	0.38596

Table 4.2 Precision-recall table at 10% compression rate

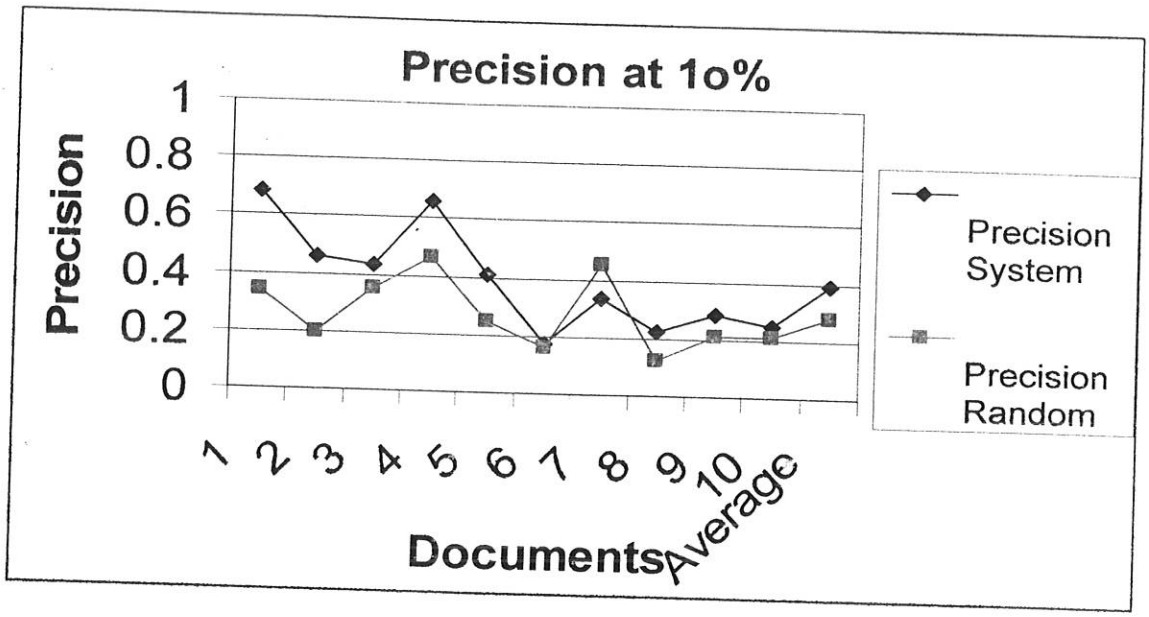

Figure 4.4 Precision graphs of system and Random summary at 10% compression rate.

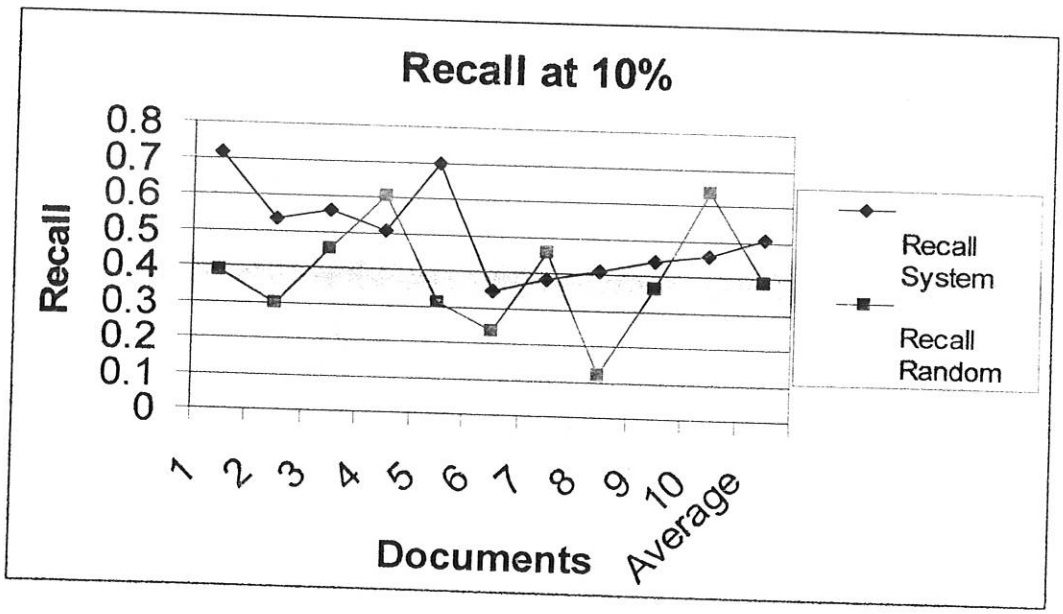

Figure 4.5 Recall graphs of system and Random summary at 10% compression rate.

From the above table and graphs one can understand that the system summary is nearest to the manual summary and its recall is also better than the random summary.

Precision of the system summary =39%

Precision of random summary =27%

Recall of system summary =50.5%

Recall of random summary =38%

When comparison is made between the results at 20% and 10% compression rate, there is significant increase in the precision but to some extent the recall decreases. Normally when the number of sentences that are extracted reduces the relevant sentences to be extracted will also reduce. This implies that recall is directly proportional to the compression rate.

4.8 Facts and Findings

On the basis of precision and recall of the system and random summary, the following analyses are made. Both types of summaries extract equal number of sentences.

At 20% compression rate out of the ten documents summarized the precision of the system summary of the seven documents are higher than the precision of the random summary. This shows that the system summary is able to extract more relevant sentences than the random summary does. On the other hand the

random summary performs better in the remaining three documents. This means the random summary's performance is lower than the system summary.

The average precision of the system summary is 33.9% whereas that of the random summary is 23%. Thus, the average result also affirms better performance of the system.

At the same compression rate the results of the evaluation made on the basis of recall indicates that out of the total documents summarized in seven of them the system summary performs well than the random summary. This indicates that the system summary is closer to the manual (ideal) summary.

The average recall of the system summary is 57% while that of the random summary is 46%. This shows that the system is closer to the manual summary than the random summary is.

Therefore, one can easily conclude that, even if the results obtained at 20% compression rate are not that much exaggerated the performance of the system is satisfactory.

To improve performance of the system it is tested at 10% compression rate. Comparison between the system summary and the random summary at this rate depicts improvement. The average precision at 20% was 33.9% and at 10% is 39% this means the systems performance is more precise at 10% than 20%.

The average precision of the random summary at 10% is 27% and that of the system summary is 39% this suggests that the system summary is able to extract more relevant sentences than the random summary. Out of the total documents considered the precision of the random summary is much lower than the system summary on 90% of the documents.

However, the average recall of the system at 10 % is lesser than the average recall at 20% this is due to the fact that recall has direct relationship with compression rate it is obvious that when compression rate decreases recall also decreases.

When comparison is made between the system and random summary with respect to their recall out of the total documents summarized on 70% of the documents the performance of the system summary is higher than that of the random summary. The average recall of the system summary is 50.5% and that of the random summary is 38 % this reaffirms better performance of the system summary.

From the above results one can understand the factors that affect the performance of the text summarizer system. Decomposing the text in to its different components has a great effect to increase the representativeness of the summary. The number and types of cue words/ phrases used and the location features also have great implications.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The goal of a summary is to produce a short representation of a long document. To do this, we need to build an abstract representation of the whole document and then generate a shorter text or select a few relevant sentences from the original text.

Automatic summarization of legal documents has a great importance to the legal experts and students, because it saves the time they spend to read unwanted details. In addition, it makes it easier to read the main ideas of many judgments in a short period of time.

This work is an initial stage in the development of automatic text summarization system for Amharic legal judgments. The work refers to the problem of processing of a huge volume of documents in the legal field, which becomes more and more difficult to access.

The method used in this research is based on extraction of relevant units in the source document by identifying the text structure and determining the themes of each segment in the decision. The summary generation is made in five stages.

The first stage is thematic segmentation: to detect the legal document structure five themes are identified, these are; Introduction, Reason, Fact, Judicial analysis and Decisions. This task is done manually.

Second, Amharic characters that have the same pronunciations are replaced by a single character. Then weight is assigned to each sentence based on its location and the cue words / phrases that it contains. After that, sentences with the highest weight are selected at 20% compression rate from each segment and merged together to serve as a single summary for the document. Finally, the system is evaluated based on intrinsic evaluation method and the results are compared against a random summary, which is automatically generated simply by taking n number of sentences from the original document.

The results obtained from the system are better when compared with the random summary. And they are good indicators.

To improve the performance sentences at 10% compression rate were extracted and the systems performance has improved to some extent. Therefore, the sentences extracted by the system summary using different extraction features are much closer to the manual (ideal) summary. Even though there are some limitations while conducting the work, the findings of this work show promising results.

The result of the conducted research shows that by using sentence extraction approach of generating summaries it is possible to automate the task of generating summaries of legal texts.

Generally this study has revealed how a computer model resembles the human task of summarizing Amharic legal texts. The promising result obtained also encourages further study in the area of automatic Amharic text summarization and lays a firm foundation for the development of a full fledged system that can work on any Amharic text.

5.2 Recommendations

The following recommendations are forwarded so as to enable future researchers in this area to upgrade the system and to make it a full-fledged work.

- The application of stemming to the original document and the cue words/phrases is highly recommended. This will reduce the effort to be exerted in order to list all possible combinations of a word suffixes and prefixes.
- The use of (Natural Language Processing) NLP methods is very important to make the extracted summary more consistent and coherent.
- The use of automatic clustering to group the different units of the legal text in to their proper thematic segments is recommended. Because it will save the time and resources, which will be spent otherwise.
- Application of task based (extrinsic) evaluation methods will have paramount importance to measure effectiveness of the system against the satisfaction of the needs of end users.
- Instead of normalizing similar alphabets a mechanism to include all the Amharic characters has to be studied.
- The existence of a well-developed Amharic legal corpus for testing the text processing systems will help a lot in measuring the performances of text processing systems in an objective way. Therefore there a standard Amharic legal corpus should be developed.
- A way to create a link between the source document and the extract (summary) has to be setup. In order to help the reader to refer a portion of the original document if he/she needs the details.

Appendix 1

The source code for the system

This system consists of the normalizer which maps similar characters in to one unique character, and the processor which performs the basic functionalities of the string processing tasks.

The input file for the system is Amharic document written using Unicode font and saved in utf-8 format

```
# -*- coding: utf-8 -*-

import codecs

import random

import sys

argc = sys.argv.__len__()

print argc

if( argc <> 4):

    exit("Syntax python helen.py inputFile  outputFile randomOutput")

else:

    fileName1 = sys.argv[1]

    outputFileName = sys.argv[2]

    randomFileName = sys.argv[3]

#fileName1 = "c:\\helen\doc5.txt"

#outputFileName = "output.doc"

#randomFileName = "random.doc"

fileName2 = "C:\\test\\cuePhrases2.txt"

keyWords = [u"ፍሬ ነገር",u"ሳቢ ብጥ",u"ምርመራ",u"ውሳኔ"]

elaborators = ["although", "though", "eventhough","but"]

import processor
```

```
import normalizer
 error = 0
 Normalizer = normalizer.normalizer()
 processor  = processor.processor()
 processor.qotationList = processor.getQotationList("C:\\test\\quotation.txt")
 try:
      inputFile  = codecs.open(fileName1,"r","utf-8")
 except:
      print "un able to open the file " + fileName1
      error = 1
if(error == 0):
      try:
           outputFile = codecs.open(outputFileName,"w","utf-8")
      except:
           print "un able to open the file " + outputFileName
           error = 1
  if(error == 0):
     cuePhrase  = []
     cuePhrase  = Normalizer.getCuePhrase(fileName2)
     line = inputFile.readline()
     sentenceWeights = []
     sentenceList = []
     AllSentenceList = []
     sentenceLength = []
     sentence = ""
```

```
sentenceWeight = 0

    maxPhraseLength = processor.getMaxPhraseLength(cuePhrase)

    compressionRate = 20

    nbSentences = 0

    SentenceWeight = 0.0

    lineIsInvalid = 0

    nbParagraph = 0

    totalNbBestSentence = 0

    counterToReduceSentenceLength = 0 #given a sentence of having N words,
counterToReduceSentenceLength tells us how many words are merged to form a phrase

                                # that should be deducted from the sentence before normalization

    #avoiding header

    currentKey = [u'ምክንያት']

    while line:

        if(line.strip() in currentKey):

            break

        line = inputFile.readline()

    line = inputFile.readline()

    #processing the remaining part of the text

    while line:

        line = line.strip()

        found = 1

        while not line and found:

            line = inputFile.readline()

            if(not line):

                found = 0
```

```
    line = line.strip()
sentenceEnd = 0
if (not line in keyWords) and found: #paragraph indicator
        if(line.find(u"::") >= 0):
            lines = line.split(u"::")
            line1 = Normalizer.normalize(lines[0])
            lines.remove(lines[0])
            line2 = "::".join(lines)
            sentenceEnd = 1
        else:
            line1 = Normalizer.normalize(line)
            line2 = ""
        if lineIsInvalid == 1:
            line1 = ""
        else:
            words = line1.split(" ")
            for elaborator in elaborators:
                end = words.__len__()
                if (elaborator in words):
                    lineIsInvalid = 1
                    pos = words.index(elaborator)
                    for i in range(pos, end):
                        words.remove(words[pos])
                    line1 = " ".join(words)
        sentence = sentence + " " + line1
```

```
            words = line1.split(" ")
            while "" in words:
                words.remove("")
            for i in range(maxPhraseLength):
                nbWords = words.__len__()
                i = maxPhraseLength - i
#               i = i + 1
                indexToRemove = []
                if(i <= nbWords):
                    for j in range(nbWords-i):
                        phrase = processor.getPhrase(words, j, i)
                        if phrase in cuePhrase:
                            sentenceWeight = sentenceWeight + 1
                            counterToReduceSentenceLength =
counterToReduceSentenceLength + (i - 1)
                            for k in range(i):
                                indexToRemove.append(j+k)
                count = 0
                for index in indexToRemove:
                    words.remove(words[index - count])
                    count = count + 1
            if(sentenceEnd == 0):   # not end of sentence
                line = inputFile.readline()
            else:
                if(line2): #sentenceEnd == 1 and next sentence continue on the line
                    line = line2
```

```
            else:
                line = inputFile.readline()
            sentence = processor.removeQotations(sentence)
            #print sentence + "\n"
            newWords = sentence.split(" ")
            while ("" in newWords):
                newWords.remove("")
            sentenceLength.append(newWords.__len__() -
counterToReduceSentenceLength)
            sentenceWeight = float(sentenceWeight)/(newWords.__len__() -
counterToReduceSentenceLength)
            counterToReduceSentenceLength = 0
            sentenceWeights.append(sentenceWeight)
            sentenceList.append(sentence)
            AllSentenceList.append(sentence)
            sentence = ""
            sentenceWeight = 0.0
            nbSentences = nbSentences + 1
            lineIsInvalid = 0
        elif line:
            N = int(compressionRate * nbSentences/100.0 + 0.99)
            totalNbBestSentence = totalNbBestSentence + N
            BestSentences = []
            sentenceWeights[0] = sentenceWeights[0] + 1.0/sentenceLength[0] #add one
to the first sentence
            sentenceWeights[nbSentences - 1] = sentenceWeights[nbSentences - 1] + 1.0
/sentenceLength[nbSentences - 1] #add one to the last sentence
            BestSentences = processor.select_N_Best(N, sentenceList, sentenceWeights)
```

```
            #outputFile.write("\n\t Current key found = " + currentKey[nbParagraph] + "\n");

            #outputFile.write("\n\tTotal number of sentences in the paragraph = " +
str(nbSentences) + "\n")

            i = 0

            #for sentence in sentenceList:

            #     outputFile.write(sentence+ "  " + str(sentenceWeights[i]) + "\n")

            #     i = i + 1

            #outputFile.write("\n\tNumber of sentences selected = " + str(N) + "\n")

            for BestSentence in BestSentences:

                        outputFile.write(BestSentence + "\n")
          currentKey.append(line)

            nbParagraph = nbParagraph + 1

            line = inputFile.readline()

            nbSentences = 0

            sentenceWeights = []

            sentenceList = []

            sentence = ""

            sentenceWeight = 0.0

      #   #processing the last paragraph

      N = int(compressionRate * nbSentences/100.0 + 0.99)

      BestSentences = []

      totalNbBestSentence = totalNbBestSentence + N

      sentenceWeights[0] = sentenceWeights[0] + 1.0/sentenceLength[0] #add one to the first
sentence

      sentenceWeights[nbSentences - 1] = sentenceWeights[nbSentences - 1] + 1.0
/sentenceLength[nbSentences - 1] #add one to the last sentence

      BestSentences = processor.select_N_Best(N, sentenceList, sentenceWeights)

      #outputFile.write("\n\t Current key found = " + currentKey[nbParagraph] + "\n");
```

```
    #outputFile.write("\n\tTotal number of sentences in the paragraph = " + str(nbSentences) +
"\n")

        #i = 0

        #for sentence in sentenceList:

        #       outputFile.write(sentence + "  " + str(sentenceWeights[i]) + "\n")

         #      i = i+1

        #outputFile.write("\n\tNumber of sentences selected = " + str(N) + "\n")

        for BestSentence in BestSentences:

              outputFile.write(BestSentence + "\n")

        line = inputFile.readline()

        nbSentences = 0

        sentenceWeights = []

        sentenceList = []

        outputFile.close()

        inputFile.close()

        nbTotalSentence = AllSentenceList.__len__()

        print "Total number of sentence = " + str(nbTotalSentence)

        print "Total number of selected sentence = " + str(totalNbBestSentence)

        selected = []

        i = 0

        while i < totalNbBestSentence:

              index = int(random.random()*nbTotalSentence)

              if(index not in selected):

                    selected.append(index)

                    i = i + 1

         try:
```

```
        outFile2 = codecs.open(randomFileName,"w","utf-8")
    except:
        print "un able to open the file " + randomFileName
        error = 1
    if(error == 0):
        for index in selected:
            outFile2.write(AllSentenceList[index])
        outFile2.close()
```

Normalizer

```
# -*- coding: utf-8 -*-
import codecs
Ha  = [u"ሀ",  u"ሃ",  u"ሐ",  u"ሓ",  u"ኀ",  u"ኃ"]
Hu  = [u"ሁ",  u"ሑ",  u"ኁ"]
Hi  = [u"ሂ",  u"ሒ",  u"ኂ"]
Hie = [u"ሄ",  u"ሔ",  u"ኄ"]
H   = [u"ህ",  u"ሕ",  u"ኅ"]
Ho  = [u"ሆ",  u"ሖ",  u"ኆ",  u"ኾ"]
Se  = [u"ሰ",  u"ሠ"]
Su  = [u"ሱ"   u"ሡ"]
Si  = [u"ሲ",  u"ሢ"]
Sa  = [u"ሳ",  u"ሣ"]
Sie = [u"ሴ",  u"ሤ"]
S   = [u"ስ",  u"ሥ"]
So  = [u"ሶ" ,  u"ሦ"]
A   = [u"አ",  u"ዐ",  u"ኣ",  u"ዓ"]
```

```
U  = [u"ኡ", u"ዉ"]

I  = [u"ኢ", u"ዊ"]

Ie = [u"ኤ", u"ዌ"]

I  = [u"እ", u"ዕ"]

O  = [u"ኦ", u"ዎ"]

TSe = [u"ጸ", u"ፀ"]

TSu = [u"ጹ", u"ፁ"]

TSi = [u"ጺ", u"ፂ"]

TSa = [u"ጻ", u"ፃ"]

TSie= [u"ጼ", u"ፄ"]

TS  = [u"ጽ", u"ፅ"]

TSo = [u"ጾ", u"ፆ"]

SHe = [u"ሸ", u"ሼ"]

GNe = [u"ኘ", u"ኜ"]

Ce  = [u"ፈጨ", u"ፌጭ"]

Che = [u"ቸ", u"ቼ"]

Wu  = [u"ዉ", u"ው"]

Je  = [u"ጀ", u"ጄ"]

class normalizer:

    """ This is designed to normalize basic normalization functionality of string processing """

    def __init__(self):

        """  """

    def getCuePhrase(self, fileName):

        """ get the list of word or phrases  """

        error = 0
```

```
lists = []
        try:
            inputFile = codecs.open(fileName,"r","utf-8")
        except:
            print "un able to find the quephrase list file " + fileName
            print "empty file is considered"
            error = 1
        if(error == 0):
            line = inputFile.readline()
            while line:
                line = line.strip()
                lists.append(line)
                line = inputFile.readline()
            inputFile.close()
            return lists
        else:
            return []
    def normalize(self,wordList):
        """ map similar chracters into one unique character """
        length = wordList.__len__()
        ModifiedString = ""
        for i in range(length):
        char = wordList[i]
        if(char in Ha):
                ModifiedString = ModifiedString  + Ha[0]
```

```
elif(char in Hu):
        ModifiedString = ModifiedString + Hu[0]
elif(char in Hi):
        ModifiedString = ModifiedString + Hi[0]
elif(char in Hie):
        ModifiedString = ModifiedString + Hie[0]
elif(char in H):
        ModifiedString = ModifiedString + H[0]
elif(char in Ho):
        ModifiedString = ModifiedString + Ho[0]
elif(char in Se):
        ModifiedString = ModifiedString + Se[0]
elif(char in Su):
        ModifiedString = ModifiedString + Su[0]
elif(char in Si):
        ModifiedString = ModifiedString + Si[0]
elif(char in Sa):
        ModifiedString = ModifiedString + Sa[0]
elif(char in Sie):
        ModifiedString = ModifiedString + Sie[0]
elif(char in S):
        ModifiedString = ModifiedString + S[0]
elif(char in So):
        ModifiedString = ModifiedString + So[0]
        elif(char in A):
```

```
            ModifiedString = ModifiedString + A[0]
    elif(char in U):
            ModifiedString = ModifiedString + U[0]
    elif(char in I):
            ModifiedString = ModifiedString + I[0]
    elif(char in Ie):
            ModifiedString = ModifiedString + Ie[0]
    elif(char in I):
            ModifiedString = ModifiedString + I[0]
    elif(char in O):
            ModifiedString = ModifiedString + O[0]
    elif(char in TSe):
            ModifiedString = ModifiedString + TSe[0]
    elif(char in TSu):
            ModifiedString = ModifiedString + TSu[0]
    elif(char in TSi):
            ModifiedString = ModifiedString + TSi[0]
    elif(char in TSa):
            ModifiedString = ModifiedString + TSa[0]
    elif(char in TSie):
            ModifiedString = ModifiedString + TSie[0]
    elif(char in TS):
            ModifiedString = ModifiedString + TS[0]
    elif(char in TSo):
            ModifiedString = ModifiedString + TSo[0]
```

```
            elif(char in Je):
                    ModifiedString = ModifiedString + Je[0]
            elif(char in SHe):
                    ModifiedString = ModifiedString + SHe[0]
            elif(char in GNe):
                    ModifiedString = ModifiedString + GNe[0]
            elif(char in Ce):
                    ModifiedString = ModifiedString + Ce[0]
                    elif(char in Che):
                    ModifiedString = ModifiedString + Che[0]
              elif(char in Wu):
                    ModifiedString = ModifiedString + Wu[0]
              else:
                    ModifiedString = ModifiedString + char
            return ModifiedString
```

Processor

```
# -*- coding: utf-8 -*-
import codecs
import re
import normalizer
Normalizer1 = normalizer.normalizer()
class processor:
    """ These is designed to implement the basic functionalities of the string processing tasks
"""
    def __init__(self):
            """ """
```

```
  self.qotationList = []
def getQotationList(self, fileName):
        lists = []
        error = 0
        try:
          inputFile = codecs.open(fileName,"r","utf-8")
      except:
          print "un able to find the quephrase list file " + fileName
          print "empty file is considered"
          error = 1
      if(error == 0):
          line = inputFile.readline()
          line = inputFile.readline()
          while line:
              line = line.strip()
              line = Normalizer1.normalize(line)
              lists.append(line)
              #print line
              line = inputFile.readline()
          inputFile.close()
        return lists
def getMaxPhraseLength(self, CuePhrase):
        """ """
        maxCount = 0
        for phrase in CuePhrase:
```

```
            lists = phrase.split(" ")
            while "" in lists:
                lists.remove("")
            count = lists.__len__()
            if(maxCount < count):
                #print lists, count
                maxCount = count
            return maxCount
    def getPhrase(self, lists, startIndex, length):
            """  get the first few words from the word list """
            phrase = ""
            for i in range(length):
#                print "index = ", i+startIndex
            phrase = phrase + lists[i + startIndex]
            phrase = phrase + " "
            phrase = phrase.strip()
            return phrase
    def removeQotations(self, sentence):
        for pattern in self.qotationList:
            sentence = re.sub(pattern,"",sentence)
        return sentence
    def  getLength(self, sentence):
        words = sentence.split(" ")
        while ("" in words):
            words.remove("")
```

```
        return words.__len__()

    def select_N_Best(self, N, sentenceList, sentenceWeights):
            """ select N sentences with the highest score  """
            bestSentence = []
            tempSentence = []
            tempIndexes = []
            tempWeights = []
            for i in range(N):
            maximum = sentenceWeights[0]
            index = 0
            length = sentenceWeights.__len__()
            for j in range(length):
                    if(sentenceWeights[j] > maximum):
                            maximum = sentenceWeights[j]
                            index = j
                    elif (sentenceWeights[j] == maximum):
                            len1 = self.getLength(sentenceList[index])
                            len2 = self.getLength(sentenceList[j])
                            if(len2 < len1):
                                maximum = sentenceWeights[j]
                                index = j
              tempIndexes.append(index)
              tempSentence.append(sentenceList[index])
            bestSentence.append(sentenceList[index])
            tempWeights.append(sentenceWeights[index])
```

```
            sentenceList.remove(sentenceList[index])
            sentenceWeights.remove(sentenceWeights[index])
        i = tempIndexes.__len__() - 1
        for index in tempIndexes:
            sentenceList.insert(index,tempSentence[i])
            sentenceWeights.insert(index,tempWeights[i])
            i = i - 1
        return bestSentence
def getSentences(self, fileName):
    error = 0
    try:
        inputFile  = codecs.open(fileName,"r","utf-8")
    except:
        print "un able to find the ile " + fileName
        error = 1
    if(error == 0):
        line = inputFile.readline()
        sentence = ""
        while line:
            line = line.strip()
            line = Normalizer1.normalize(line)
            sentence = sentence + line
            line = inputFile.readline()
        inputFile.close()
        return sentence
```

```
            else:
                    return "@"
# -*- coding: utf-8 -*-
import processor
processor  = processor.processor()
import sys
argc = sys.argv.__len__()
print argc
if( argc <> 3):
      exit("Syntax python helen.py manualSummaryFile  random/SystemOutput")
else:
      fileName1 = sys.argv[1]# manual summary
      fileName2 = sys.argv[2] #system/random output file
#fileName1 = "C:\\Helen\\manualoutput.txt"
#fileName2 = "C:\\Helen\\systeOutput.txt"
sentence1 = []
sentence2 = []
sentence1 = processor.getSentences(fileName1)
sentence2 = processor.getSentences(fileName2)
if(sentence1 <> "@" and sentence2 <> "@"):
      words1 = sentence1.split(" ")
      words2 = sentence2.split(" ")
      while ("" in words1):
            words1.remove("")
      while ("" in words2):
```

```
    words2.remove("")
TotalRetrieved = words2.__len__()
TotalRelevant = words1.__len__()
relevantRetrived = 0.0
for word in words1:
        if word in words2:
                relevantRetrived = relevantRetrived + 1.0
precision = relevantRetrived/TotalRetrieved
recall = relevantRetrived/TotalRelevant
print "precission = " + str(precision)
print "Recall = " + str(recall)
```

References

1. Baye, Y. (1997): የ አማሪኛ ሰዋሰው። አዲስ አበባ ዩኒቨርስቲ። አዲስ አበባ
2. Beletu, R (1982): Graphemes Analysis of Writing System of Amharic. Paper for the Requirement of Bachelor of Art in Linguistics. Addis Ababa University. Addis Ababa.
3. Bender and Fulass (1976): The Ethiopian Writing System. In languages in Ethiopia: Oxford University Press. London.
4. Bethelehem, M. (2002): N-gram based Automatic Indexing for Amharic Text. (Masters Thesis) School of Information Studies for Africa. Addis Ababa University. Addis Ababa.
5. Bizuneh, M. (2003): The Application of WEBSOM for Amharic Text Retrieval. (Masters Thesis) Faculty of Informatics. Addis Ababa University. Addis Ababa
6. Chowdhury, G.G. (1997): Introduction to modern Information Retrieval. Addis Ababa University. Addis Ababa.
7. Dalianis, H. (2000): SweSum. Text summarization for Swedish URL: http://www.nada.kth.se/~hercules/Textsumsummary.html
8. Dalianis, H. et al (2002): Automatic text summarization for Scandinavian languages.
   URL:http://www.nada.kth.se/~hercules/scandsum/ScandSumArsbog2002.pdf
9. Dalianis, H. (2004): What is automatic text summarization? http://www.dsv.su.se/~hercules/textsammanfattningeng.html

10. Dalianis, H. Hassel, M. etal (2002): From Swesum to Scandsum. Automatic text summarization for the Scandinavian languages.

11. Dalianis, H. (2001): Automatic text generation and summarization URL:http://www.nada.kth.se/~hercules/Textsumsummary.html.

12. Daniel, G. (2003): An Integrated Approach to Automatic Complex Sentence Parsing for Amharic Text. (Masters Thesis) Faculty of Informatics. Addis Ababa University. Addis Ababa.

13. Edmundson, H.P. (1969): New Methods in Automatic Extraction Journal of the ACM 16(2) pp 264-285.

14. Ethiopia, T. (2002): Application of Case-Based Reasoning for Amharic Legal Precedent Retrieval: A case Study with the Ethiopian Labor Law. Masters Thesis. Addis Ababa University. Addis Ababa.

15. Farzindar, A. and Lapalme G. (2004): Legal texts summarization by exploration of the thematic structures and argumentative roles. ACL-2004 Text Summarization Branches Out Workshop, 2004 - acl.ldc.upenn.edu

16. Farzindar, A. and Lapalme G. (2004): The Use of Thematic structure and Concept Identification for Legal Text Summarization URL: http://www.iro.umontreal.ca/~farzinda/FarzindarCLINE04.pdf

17. Farzinder, A. and Lapalme G. (2004). LetSum, an Automatic Legal Text Summarizing System

URL: http://acl.ldc.upenn.edu/acl2004/textsummarization/pdf/Farzindar.pdf

18. Fen, L. S. D John. D. T. (2004): Feature Selection for Summarizing: The Sunderland

URL:http://wwwnlpir.nist.gov/projects/duc/pubs/2004papers/usunderland.liang.pdf

19. Francine, M. (2000): Automatic Indexing and Abstracting of Document texts. Kluwer academic publishers.

20. Getachew, H. (1967): The Problems of Amharic Writing System. Addis Ababa University. Addis Ababa.

21. Goldstein, J. et al (1999): Summarizing Text Documents: Sentence Selection and Evaluation Metrics. Carnegie Mellon University. USA

22. Grover, C. et al (2003): Summarizing Legal Texts: Sentential Tense and Argumentative Roles. URL: http://acl.ldc.upenn.edu/W/W03/W03-0505.pdf

23. HLT-NAACL 2003 Workshop: Text Summarization (DUC03), 2003 - acl.ldc.upenn.edu

24. Hahn, U. & Mani, I. (1999): The goal of automatic summarization URL:http://www.dsv.su.se/ijcai-99/tutorials/e3.html

25. Hu.p et al (2004): Chinese Text Summarization Based on Thematic Area Detection.

URL:http://www.acl.ldc.upenn.edu/acl2004/textsummarization/pdf/Hu.pdf

26. Hutchins, J. (1993): Introduction to"Text Summarization" Workshop. University of East Anglia, Norwich, UK.

27. Jing et al (1998): Summarization Evaluation Methods : Experiments and Analysis. URL: ftp://ftp.cs.bgu.ac.il/pub/people/elhadad/sum-eval.ps.gz.

28. Kamil Nuru (2005): Automatic Amharic News Text Summarizer. (Masters Thesis). Faculty of Informatics. Addis Ababa University. Addis Ababa.

29. Kupiec, J. etal (1995): A trainable document summarizer. Xerox Palo Alto research center California.

URL=http://www.dcs.shef.ac.uk/~mlap/teaching/kupiec95trainable.pdf

30. Lancaster, f.w. (1998): Indexing and Abstraction in theory and practice. Library Association Publishing, London.

31. Lin, Y.C and Hovy, E. (2002): AUTOMATED TEXT SUMMARIZATION AND THE SUMMARIST SYSTEM

32. Luhn, H.P. (1958): The Automatic Creation of Literature Abstract. IBM Journal of Research and Development. Pp159-165.

URL:http://www.research.ibm.com/journal/rd/022/luhn.pdf.

33. Mani, I. (2001): Summarization Evaluation: An Overview. URL: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings2/sum-mani.pdf

34. McDonald, D. (2002): Using Sentence-Selection Heuristics to Rank Text Segments in TEXTRACTOR. University of Arizona

35. Meadow, J et al (2000): Text Information Retrieval systems. Academic press.

36. Minel L.J et al (1997): How to Appreciate the Quality of Automatic Text Summarization. EACL, Workshop Intelligent scalable Text Summarization pp25-31. Madrid Spain

37. Moens et al, (1999): Abstracting of legal cases: the potential of clustering based on the selection of representative objects. Journal of the American Society for Information Science, 50(2):151–161.

47. Teufel, S. Moens, M. (2002): Summarizing scientific articles - experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445.

48. The American law institute (1974): Corpus Juris Secundum vol 49. pp 87-113.The American law book and West Publishing Company.

49. Turney, p. (1997): Extraction of Key phrases from Text: Evaluation of Four Algorithms: National Research Council of Canada

50. Wantanabe, H. (1996): A Method for Abstracting Newspaper Articles by Using Surface Clues. Tokyo Research Laboratory.

51. Wong. L. (1998): Automatic news summarization and extraction system. MEng Computing. URL: http:// www. doc. ic. ac uk/and/ surprise – 971 journal/vo14/ks/sum html.

DECLARATION

This thesis is my original work, hasn't been presented for a degree in any other university and all that all sources of material used for the thesis have been duly acknowledged.

_____

HELEN ADANE TEKLE

2006

The thesis has been submitted for examination with my approval as university advisor.

_____

Dr. NEGA ALEMAYEHU

47. Teufel, S. Moens, M. (2002): Summarizing scientific articles - experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445.

48. The American law institute (1974): Corpus Juris Secundum vol 49. pp 87-113.The American law book and West Publishing Company.

49. Turney, p. (1997): Extraction of Key phrases from Text: Evaluation of Four Algorithms: National Research Council of Canada

50. Wantanabe, H. (1996): A Method for Abstracting Newspaper Articles by Using Surface Clues. Tokyo Research Laboratory.

51. Wong. L. (1998): Automatic news summarization and extraction system. MEng Computing. URL: http:// www. doc. ic. ac uk/and/ surprise – 971 journal/vo14/ks/sum html.

DECLARATION

This thesis is my original work, hasn't been presented for a degree in any other university and all that all sources of material used for the thesis have been duly acknowledged.

HELEN ADANE TEKLE

2006

The thesis has been submitted for examination with my approval as university advisor.

Dr. NEGA ALEMAYEHU