Tribhuvan University
Institute of Science and Technology

Automatic Text Summarization System for Nepali Language Based on Sentence Extraction

Dissertation
Submitted to

Central Department of Computer Science & Information Technology
Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements
for the Master's Degree in Computer Science & Information Technology

By
Rajendra Lamichhane
December, 2013

Tribhuvan University
Institute of Science and Technology

Automatic Text Summarization System for Nepali Language Based on Sentence Extraction

Dissertation
Submitted to

Central Department of Computer Science & Information Technology
Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements
for the Master's Degree in Computer Science & Information Technology

By
Rajendra Lamichhane
December, 2013

Supervisor
Prof. Dr. Shashidhar Ram Joshi

Co-Supervisor
Mr. Bikash Balami

Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science & Information Technology

Student's Declaration

I hereby declare that I am the only author of this work and that no sources other than the listed here have been used in this work.

... ...
Rajendra Lamichhane
Date: 29 December, 2013

Supervisor's Recommendation

I hereby recommend that this dissertation prepared under my supervision by **Mr. Rajendra Lamichhane** entitled **"Automatic Text Summarization System for Nepali Language Based on Sentence Extraction"** in partial fulfilment of the requirements for the degree of M.Sc. in Computer Science and Information Technology be processed for the evaluation.

... ...
Prof. Dr. Shashidhar Ram Joshi
Department of Electronics & Computer Engineering,
Institute of Engineering,
Pulchowk, Nepal

Date: 29 December, 2013

Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science & Information Technology

LETTER OF APPROVAL

We certify that we have read this dissertation and in our opinion it is satisfactory in the scope and quality as a dissertation in the partial fulfillment for the requirement of Masters Degree in Computer Science and Information Technology.

Evaluation Committee

... ...
Asst. Prof. Nawaraj Paudel
Central Department of Computer Science & Information Technology,
Tribhuvan University, Kathmandu, Nepal
(Head)

... ...
Prof. Dr. Shashidhar Ram Joshi
Department of Electronics & Computer Engineering, Institute of Engineering,
Pulchowk, Kathmandu, Nepal
(Supervisor)

... ...
Asst. Prof. Lalita Sthapit
(External Examinar)

... ...
Mr. Arjun Singh Saud
(Internal Examinar)

Date: 28 January, 2014

ACKNOWLEDGEMENTS

Though only my name appears on the cover of this dissertation, a great many people have contributed for its completion. I owe my gratitude to all those people who have made this dissertation possible and because of whom my graduate experience has been one that I will cherish forever.

My deepest gratitude is to my respected teacher and dissertation advisor **Prof. Dr. Shashidhar Ram Joshi**, Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk, for providing me invaluable suggestions, encouragement and strong guidelines throughout this research period. Without his cooperation and suggestion, this dissertation could not be completed. With this regard, I wish to extend my sincere appreciation to respected Head of the Central Department of Computer Science and Information Technology (CDCSIT), **Asst. Prof. Nawa Raj Poudel** for his kind help, encouragement and constructive suggestions regarding the dissertation.

My co-advisor, Mr. Bikash Balami, has been always there to listen and give advice. I am deeply grateful to him for the long discussions that helped me to sort out the technical details of my work. Without his guidance, invaluable co-supervision and continuous encouragement, this dissertation would never have come in this form.

I am very grateful and thankful to all the respected teachers of CDCSIT, TU, for granting me broad knowledge and inspirations within the time period of two years.

My special thanks goes to Mr. Ashok Kumar Pant for his kind help in implementation of some problems as well as in corrections of the document. Also, I am thankful to Mr. Okil Dhakal and my brother Mr. Rajiv Lamichhane for their help in collecting and annotating the data and generating the manual summary.

Likewise, I would like to thank to my colleagues and friends Mr. Pravakar Ghimire, Mr. Roshan Silwal, Mr. Dinesh Kuamar Khadka, Mr. Deepednra Bhatt, Mr. Surya Bam who directly and indirectly extended their hands in making this thesis work successful and complete.

As we know that, there won't be 100% accuracy and efficiency in any work done either by machine or human. So there may occur some errors in my project. But I have done my best to complete this dissertation. So any suggestion regarding the mistakes of this work will be always welcomed.

Rajendra Lamichhane

29 December, 2013

ABSTRACT

Automated text summarization is a generic problem in the Natural Language Processing (NLP) community. It has grabbed great attention recently as the amount of information increases throughout the world, online and offline. As the volume and availability of data increases, it causes redundancy and scatterness over the world. So, there is the need of effective and powerful tool to summarize text documents automatically. So far, many researches have been done for English and other European languages with high performance. However, Nepali language still suffers from the little attentions and researches in this field.

In this dissertation, a method has been proposed, which lets us to summarize Nepali text documents automatically based on sentence extraction techniques. The various stages involved in this approach which are: text preprocessing, feature extraction, sentence scoring and ranking, and summary generation. The proposed system is tested with various datasets collected from different sources such as books, newspapers, article, reports, etc. Automated evaluation techniques are used to validate the proposed system against the manual summaries. The overall accuracy of the proposed system is achieved as 79.18% precision, 71.77% recall and 75.02% F-Score. Cosine similarity measure gives overall similarity of 91.16% between manual summary and system summary.

Keywords:

Automated text summarization, Natural language processing, Nepali language, Preprocessing, Feature extraction

TABLE OF CONTENTS

Acknowledgement i

Abstract ii

List of Figures v

List of Tables vi

Abbreviations vii

1 INTRODUCTION 1
- 1.1 Introduction . . . 1
- 1.2 Classification of Text Summarization Task . . . 2
- 1.3 Applications of Automatic text Summarizations . . . 4
- 1.4 Motivation . . . 5
- 1.5 Background of Nepali NLP . . . 5
- 1.6 Challenges . . . 6
- 1.7 Problem Definition . . . 6
- 1.8 Objectives . . . 7
- 1.9 Contribution of the Thesis . . . 7
- 1.10 Outline of the Document . . . 7

2 LITERATURE REVIEW 8
- 2.1 Single Document Summarization Approaches . . . 8
- 2.2 Multiple Document Summarization Approaches . . . 10
- 2.3 Summarization in Nepali Language . . . 11

3 RESEARCH METHODOLOGY 12
- 3.1 System Overview . . . 12
- 3.2 Data Acquisition . . . 13
- 3.3 Preprocessing . . . 13
  - 3.3.1 Symbols and Punctuation Marks Removal . . . 14
  - 3.3.2 Stop Words Removal . . . 14
  - 3.3.3 Stemming . . . 14
- 3.4 Feature Extraction . . . 15
  - 3.4.1 Length of Sentence . . . 15
  - 3.4.2 Position of Sentence . . . 15
  - 3.4.3 Named Entity Recognition . . . 16
  - 3.4.4 Term Frequency . . . 16
- 3.5 Sentence Evaluation . . . 17

3.6 Summary Generation . . . 17
3.7 System Evaluation Metrics . . . 17
3.7.1 Precision/Recall/F-Score . . . 17
3.7.2 Cosine Similarity . . . 18

4 IMPLEMENTATION 20
4.1 Document Preprocessing Algorithms . . . 20
4.1.1 Symbols and Punctuation Marks Removal Algorithm . . . 20
4.1.2 Stop Words Removal Algorithm . . . 20
4.1.3 Stemming Algorithm . . . 21
4.2 Summary Generation Steps Demonstration . . . 21
4.2.1 Input Document . . . 21
4.2.2 Symbosl and Punctuation Marks Removal . . . 22
4.2.3 Stop Word Removal . . . 23
4.2.4 Stemming . . . 23
4.2.5 System Summary . . . 23
4.2.6 Mannual Summary . . . 24
4.3 Document Feature Vector and Sentence Ranking . . . 24

5 EXPERIMENTATIONS AND RESULTS 26
5.1 Testing Datasets . . . 26
5.1.1 Dataset 1: Book . . . 26
5.1.2 Dataset 2: Kantipur News . . . 26
5.1.3 Dataset 3: Himalkhabar Patrika . . . 27
5.1.4 Dataset 4: Nagarik News . . . 27
5.1.5 Dataset 5: Online Khabar Patrika . . . 27
5.2 Data Dictionaries . . . 27
5.2.1 Symbols and Punctuation Marks Dictionary . . . 27
5.2.2 Stop Word Dictionary . . . 27
5.2.3 NE Dictionary . . . 28
5.3 Experimentation Results . . . 29
5.3.1 Experiment 1 . . . 29
5.3.2 Experiment 2 . . . 31
5.3.3 Experiment 3 . . . 33
5.3.4 Overall System Performance and Result Analysis . . . 35

6 CONCLUSION 37
6.1 Conclusion . . . 37
6.2 Limitations and Future Scope . . . 37

Appendix A Sample Source Codes 42

Appendix B Sample Input and Output 57

LIST OF FIGURES

1.1 Classification of Summarization Tasks. 3

3.1 Top Level System Model. 12
3.2 Detail Architecture of System Model. 13
3.3 Some List of Nepali Stop Words. 14

4.1 Sample of Input Document. 22
4.2 Symbols and Punctuation Marks Removed Output. 22
4.3 Stop Word Removed Output. 23
4.4 Stemming Output. 23
4.5 System Summary Output. 24
4.6 Manual Summary. 24

5.1 Symbol and Punctuation Marks Dictionary. 27
5.2 Stop Word Dictionary. 28
5.3 Precision-Recall-FScore Graph of Experiment 1. 30
5.4 Similarity Graph of Experiment 1. 31
5.5 Precision-Recall-FScore Graph of Experiment 2. 32
5.6 Similarity Graph of Experiment 2. 33
5.7 Precision-Recall-FScore Graph of Experiment 3. 34
5.8 Similarity Graph of Experiment 3. 35
5.9 Graph of Overall Precision-Recall-FScore-Similarity. 36

LIST OF TABLES

4.1 Document Features and Sentence Ranking. 25

5.1 NE Dictionary 29
5.2 Result of Experiment 1. 30
5.3 Result of Experiment 2. 32
5.4 Result of Experiment 3. 34
5.5 Average of System Measures. 35

LIST OF ABBREVIATIONS

ATS Automatic Text Summarization

ATSSNL Automatic Text Summarization System for Nepali Language

IDF Inverse Document Frequency

IR Information Retrieval

JDK Java Development Kit

MS Manual Summary

NDS Number of Document Sentence

NLP Natural Language Processing

NMSS Number of Manual Summary Sentence

NNLP Nepali Natural Language Processing

NSSS Number of System Summary Sentence

P Precision

PDA Personal Digital Application

POS Part of speech

R Recall

SS Summarization System

TF Term Frequency

TF-IDF Term Frequency x Inverse Document Frequency

TF-ISF Term Frequency x Inverse Sentence Frequency

TS Text Summarization

Chapter 1

INTRODUCTION

1.1 Introduction

In this age of electronic and communication, the information in the internet rapidly grows by huge amount day by day and we cannot retrieve the required document or information easily and quickly. The concept of text summarization was introduced in English and other European language in the era of the fifties. Text summarization has become an important and timely tool for assisting and interpreting text information in todays age. It is very difficult for human beings to manually summarize large documents of text. There is an copiousness of text material available on the Internet. However, usually the Internet provides more information than is needed. Therefore, a twofold problem is encountered: searching for relevant documents through an overwhelming number of documents available, and absorbing a large quantity of relevant information. The goal of automatic text summarization is condensing the source text into a shorter version preserving its information content and overall meaning.

Automatic Text Summarization (ATS) is the process of reducing the given text document into a summary that retains the most important points of the original document. It is a process to produce an abstract by selecting a significant portion of the information from one or more documents by preserving its information contents and overall meaning. In an automatic text summarization process, a text is given to the computer and the computer returns a shorter less redundant extract or abstract of the original text.

The summary is the condensed representation of a document's contents. A summary outlines important aspects of the document in a precise way. It should be informative and providing the most important information in the document. A summary should be non repetitive and as brief as possible. In a text, the same information can be repeated to emphasize its importance, but a summary should give as much precise information as possible. A summary should be indicative, it should indicate the document's relevance to the reader.

Humans write a document to represent an idea, event or an opinion. Text evolves around a general concept, which is coherently partitioned into sub topics, that supports the main topic. The summary should capture the general idea and should include important topics.

According to Eduard Hovy, the summary can be defined formally as in [1]:
A summary is a text that is produced from one or more texts, that contains a significant portion of the information in the original texts, and that is no longer than half of the original text.

This simple definition captures following three important aspects that characterize research on automatic summarization:

1. Summaries may be produced from a single document or multiple documents either in abstract or extract way,
2. Summaries should preserve important information of the document,
3. Summaries should be short as far as possible.

The Text Summarization (TS) methods can be classified into extractive and abstractive summarization [2]. An extractive summarization method consists of selecting important sentences, paragraphs etc. from the original document/text and concatenating them into a shorter form to make a summary. The importance of sentences is decided based on statistical and linguistic features of sentences. An abstractive summarization attempts to develop an understanding of the main concepts in a document and then express those concepts in clear natural language. It uses linguistic methods to examine and interpret the text and then to find the new concepts and expressions to best describe it by generating a new shorter text that conveys the most important information from the original text document. That is, the summary containing sequence of words not present in the original document.

Summarization is a hard problem of Natural Language Processing because, to do it properly, one has to really understand the point of a text. This requires semantic analysis, discourse processing, and inferential interpretation. The last step, especially, is complex, because systems without a great deal of world knowledge simply cannot do it. Therefore, attempts so far of performing true abstraction, creating abstracts as summaries, have not been very successful.

1.2 Classification of Text Summarization Task

The goal of automatic summarization is the construction of a concise and coherent summary of one or several documents. The task of automatic summarization can vary depending on several criteria such as the type of the produced summaries, the number of source documents, the use of external resources and the task specific constraints. The diagram in Figure 1.1 shows different kinds of summarization tasks depending on the mentioned criteria. The rest of this subsection explains each of the shown tasks. As mentioned in the introduction section above, a summary can either be an abstract or an extract. Depending on whether we want to produce an abstract or an extract summary, the summarization process will be abstraction based or extraction based respectively.

Generation of a summary of one document is often referred to as single document summarization while of several documents as multi document summarization. Multi document summarization is obviously a more complex task than the single document summarization. There

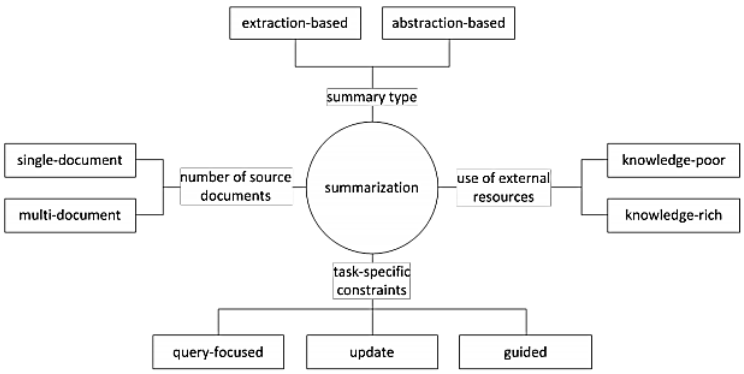

Figure 1.1: Classification of Summarization Tasks.

are two major reasons for this. First, information overlap between the documents can lead to redundancy in the summary. Secondly, an extra effort is required to organize the information from several documents to a coherent summary.

Another criterion for classification of summarization methods is the use of external resources. Knowledge-poor techniques don't use any external resources while knowledge-rich techniques may utilize external corpus such as Wikipedia or lexical resources such as WordNet. Such resources are often used to unravel semantic relations between words, phrases or sentences. There are several extensions of a classical summarization task. In query-focused or query oriented summarization a query is provided to a summarizer in addition to the source documents. The summarizer is supposed to construct a summary that contains information requested by the query. A document retrieval system together with a query-oriented multi document summarization system is potentially a very powerful combination, which might be much more effective than a document retrieval system alone. Another extension is the so called update summarization. The idea of update summaries comes from the experiments with summarization of news articles that are chronologically organized. The purpose of the update summary is to identify new pieces of information in the more recent articles with the assumption that the user has already read the previous ones.

Very recent extension of the summarization task is the guided summarization presented in [3]. In guided summarization a set of aspects that should be covered in a summary is provided. The documents are classified into several categories such as accidents, natural disasters, criminal attacks etc. Aspects vary depending on the category. For example, the summary of documents related to an accident should cover the following aspects: what happened, data, location, reason for the accident, casualties, damages and rescue efforts. The motivation for guided summarization is to inspire innovative approaches that rely on deeper linguistic analysis rather than superficial statistical features such as term frequencies

1.3 Applications of Automatic text Summarizations

For any area, automatic text summarization is a very powerful tool to save time and resources, and optimizing availability. Text summarization can be used for:

- saving time and resource: If we need to generate the summary of a large number of documents of any types then it cannot be possible by single resource with in less time.
- speeding up the information retrieval and text mining processes.
- displaying text on handheld devices, such as PDAs.

Automatic text summarization can be useful in many fields such as:

- **Medical area:** A lot of documents are published in medical research in the last two decades, and in many cases a medical specialist is in a deep need to find relevant information about patient‘s conditions timely. So, text summarization here saving time resources and optimizing the availability of medical experts.
- **Legal area:** Legal resources and documents are sparse and expensive in time and expertise level, and cost, which yields legal experts perform difficult and responsible work. Thus automatic text summarization needed for expert to be able to find compressed and restated content of relevant judicial documents, including laws and their proposals, relevant court decisions or tribunal process summarizations [4].
- **News area:** Thousands of political, sport, economic and other types of news are published every second on the internet. It‘s very hard or impossible to browse all of them or either the half. By using text summarization user can find which news she/he is concerned with before reading the whole text.

On the internet, one can find a lot of examples of automatic text summarization systems as:

- Google news, Microsoft news, Columbia news blaster which returns an abstracted summary for news in the world.
- Blog summarization tool and aggregation and opinion survey systems.
- Word Summarizer: this also included in Microsoft. Word.
- Personal Digital Application (PDA) which contains summarizer applications.

1.4 Motivation

The information available on internet, organization, university library, office, etc. are abundant. Usually, the number of relevant queries to search appropriate document from the user is humongous which intermingle with the irrelevant documents. Each retrieved document could be very lengthy and readers find that it is very tedious and time consuming to condense the main gist of the documents. Although, a number of tools (i.e. MS AutoSum, Summarist, etc.) is available to facilitate the text summarization process automatically but these are mainly for English and other European language but not for Nepali language. With the rapid increase of local language contents in electronic form and the gradual improvements of Nepali language resources for computational models, the possibility of developing some language processing applications for Neplai has increased. An automatic text summarizer is one such major application which many people can take benefit from it because it helps people to get the most important and relevant information in a shorter time. The motivation of developing an automatic text summarizer for Nepali is empowered under these circumstances. Automatic text summarization in Nepali language is very difficult and challenging and there is no any works have been done for Nepali till this work was started.

1.5 Background of Nepali NLP

Natural Language Processing (NLP) is a new field for research and development in Nepal. The first NLP works in Nepal include the Nepali Spell Checker and Thesaurus that got released in the year 2005. The Spell Checker and the "Dobhase", an English to Nepali machine translation projct respectively developed in collaboraion by Madan Puraskar Pustakalaya (MPP, http://www.mpp.org.np) and Kathmandu University (http://ku.edu.np). In the succeeding year, further works on language engineering like corpus building and annotation for Nepali, Text-To-Speech System for Nepali, digitized Nepali dictionary also got started under the NelRaLEC (Nepali Language Resources and Localization for Education and Communication) Project, also known as the Bhasa Sanchar Project (http://www.bhasasanchar.org) and currently being run at Madan Puraskar Pustakalaya Nepal [5].

With the above mentioned NLP tools developed, further doors of possibilities would open up for the research and development of several useful Natural Language Processing applications Machine Translation systems (currently we just have a unidirectional one from English to Nepali), Question Answering systems, Grammar Checker for Nepali, Information Retrieval Systems, Expert Systems and so on and so forth. Given the socio-economic realities and constraints of Nepali and Nepalese, such a substantial growth in the focus towards the research and development of NLP applications is certain to bring about some positive impacts, some of which include a concrete contribution towards bridging the existing digital divide and the development of the expertise in the local language computing.

Also, some of the research work has been done in Central Department of Computer Science and Information Technology (CDCSIT, http://www.cdcsit.edu.np) in the subject of Nepali Natural Language Processing (NNLP). The works such as “A Chunk Level Statistical Machine Translation (English Language to Nepali Language Translation)”, “Creation of Parallel Corpus from Comparable Corpus (For English-Nepali Language Pair)”, "Named Entity Recognition", etc has been done in the field of Nepali NLP.

1.6 Challenges

Text summarization has been a research topic since the 1950s, however, it has been became more active till now. The aim of text summarization research can be said to obtain a good summary or summaries, but it has been thought difficult to produce good summary as generated by human i.e, abstract and evaluate it because we do not have a definite standard measures to evaluate such systems. Many researchers were compared the summary obtained from automatic text summarization to human reference summary. Since, the two human reference summary may differ for the same documents so the evaluation system for the automatic text summarization is very difficult. Few other challenges could be, word sense ambiguity, this is the ambiguity created sometimes due to abbreviations that can have more than one acronym. Here, depending on the subject we have to match the acronym for better understanding. Another challenge, is interpreting long sentences and jargons. Usually, the flow of information in a given document is not uniform, which means that some parts are more informative than others. The major challenge in summarization lies in distinguishing the more informative parts of a document from the less ones. Though there have been instances of research describing the automatic creation of abstracts, most work done on automatic text summarization relies on verbatim extraction of sentences to address the problem of single document summarization.

In the case of Nepali language, it is a resource poor language annotated corpora, name dictionaries, good morphological analyzers, POS taggers, stemming etc. are not yet available in the required measure. Although Nepali language have a very old and rich literary history, technological development are of recent origin. Web sources for name lists are available in English, but such lists are not available in Nepali forcing the use of transliteration for creating, such lists.

1.7 Problem Definition

In this era, human beings have so busy life and they don’t have much time to read large documents/reports but sometimes they have to present about the report in some where at any point of time. If such people get the summary version of such documents then obviously they have easy to present about reports. But, to generate a summary of a large document is difficult, tedious and time consuming task itself for human beings so the automatic summarization of the document is needed. Since, there is no system yet that can be used to generate the summary for the

Nepali text documents, it is still very time consuming and tiresome work to go through all the contents line by line and get the summary or theme of the document. For large volume of data, information generation by manual readers can not be feasible. It required much time and effort as domain diversity increases. So, machine learning strategies are required for automated text summarization. There needs a learning model that can automatically generate the summary of the given Nepali text documents, and can perform analysis on accuracy of machine generated summary with human generated summary.

1.8 Objectives

The main objective of this research work is to propose a learning model that can perform automatic Nepali text summarization for single document. The proposed model can be used in various domains with recognizing novelty and ensuring that the final summary is both coherent and complete. More clearly, following list describes the research objectives.

1. To propose a automated learning and summary extraction model for Nepali text documents.

2. To analyze the accuracy of proposed model against human performance.

1.9 Contribution of the Thesis

The main contribution of this thesis is the identification of general steps of an automatic extractive text summarization method for single document for Nepali language. That is, this dissertation provides the statistical approach to find the summary of the given documents. This thesis also explains about how we can validate our system automatically.

1.10 Outline of the Document

The remaining part of the document is organized as follows,

Chapter 2 describes necessary background information and related work of summarization research on single document as well as in multi document.

Chapter 3 defines in detail the system structure and the proposed approach including the corpus collection and annotation details, feature selection and extraction and the used classification approaches (scoring approach).

Chapter 4 describes the implementation details of the system. All the methods described in the Chapter 3 are implemented for system evaluation.

Chapter 5 includes experimentation results in tabular as well as in graphs. There will be two graphs for each experiment, and also for the overall average for the sysem is shown.

Chapter 6 concludes the system performance and future directions.

Chapter 2

LITERATURE REVIEW

This chapter consists of a brief study of the technical background behind the automatic summarization and the comprehensive study carried out so far to find the current status of the field. The different approaches that have been taken over the last six decades to automatically summarize single text document as well as multiple text document will be explained and the applicability of such approaches for less resourced languages such as Nepali will be discussed in this chapter.

2.1 Single Document Summarization Approaches

Father of Information Retrieval, Hans Peter Luhs in 1958 proposed the first approaches of the automatic text summarization to generate a summary for the document. In his proposed system he used the frequency of a particular word as a measure of sentence significant in [6]. He derived a significant factor that reflects the number of significant word's occurrence within a sentence of a document, and the linear distance between them due to the intervention of none significant words. After that, all of these sentences in the document are ranked and the top most ranked sentences are selected to include in abstract summary.

Many previous works on extractive summarization includes two major steps: first step is ranking the sentences based on their scores which are computed by combining few or all of the features such as term frequency (TF), positional information and cue phrases in [7] also in [8] and the second step is selecting a few top ranked sentences to form an extract. The very first work on automatic text summarization computes the salient sentences based on word frequency (number of times a word occurs in a document) and phrase frequency. Although subsequent research has developed sophisticated summarization methods based on various new features, the work presented in [9] is still followed today as the foundation for extraction based summarization.

A text summarization technique using extracted keywords is proposed in [10]. It describes four stages of summarization. The first stage is preprocessing stage, which converts the unstructured text into structured by removing the stop words, parsing the text and assigning the Part of speech (POS) tag for each word in the text and store the result in a table. The second stage is to extract the important key-phrases from the text by selecting the candidate words. The system uses the extracted keywords/key-phrases to select the important sentence. Each sentence ranked

depending on many features such as the existence of the keywords/key-phrase in it, the relation between the sentence and the title by using a similarity measurement and other many features. The third stage of the proposed system is to extract the sentences with the highest rank. The fourth stage is the filtering stage, where the relevant sentences are filtered.

Text summarization using sentence extraction for the Bengali language is presented in [11]. The proposed system based on sentence ranking mechanism. Each sentence in the document are ranked based on their weight after some preprocessing is performed. The preprocessing stage includes the removal of stop words, stemming and breaking the input document into a collection of sentences. In the next stage, sentence ranking is performed based on their scores using thematic term, positional value and sentence length calculation. A summary is generated by selecting k-top ranked sentences. To increase the readability of the summary, the sentence in the summary are reordered based on their appearance in the original documents.

Also, an automatic text summarization based on semantic feature extraction for the Arabic language is presented in [12]. The proposed model is mainly focused on preprocessing and feature extraction of sentence and ranking them based on the score they have. It also focused on semantic analysis of the words of the sentence. Various steps are performed in the approach, some of them are preprocessing of text, extract a set of feature from sentences, classify sentences based on the scoring method, ranking sentences and finally generate an extract summary.

Work done in [13] describes the state of the art of automated text summarization techniques for a single document. It compares most of the recent techniques used in text summarization. Similarly [14] describes the various techniques of automated multiple document summarization and made comparison among them.

The paper [15] presents a sentence reduction system for automatically removing extraneous phrases from sentences that are extracted from a document for summarization purpose. The system uses multiple sources of knowledge to decide which phrases in an extracted sentence can be removed, including syntactic knowledge, context information, and statistics computed from a corpus which consists of examples written by human professionals. Reduction can significantly improve the conciseness of automatic summaries.

A practical approach for extracting the most relevant sentences from the original document to form a summary is presented in work [4]. A proposed approach takes advantages of both the local and global properties of sentences. The algorithm that combines these properties for ranking and extracting sentences is given. The local property can be considered as clusters of significant words within each sentence, while the global property can be thought of as relations of all sentences in a document.

The efficient technique for language independent generic extractive summarization for single document is presented in [16]. The algorithm is based on structural and statistical (rather than semantic) factors. Through evaluations performed on a single document summarization for English, Hindi, Gujarati and Urdu documents, the method performs equally well regardless

of the language. The algorithm has been applied on DUC data for English documents and various newspaper articles for other languages with corresponding stop word list and modified stemmer. The results of summarization have been compared with DUC 2002 data using degree of representativeness. For other languages, the degree of representativeness we get is highly encouraging.

2.2 Multiple Document Summarization Approaches

When a user query about a topic, hundreds of documents are returned to him. If we deal with each document alone and summarized it, then hundred of summaries are generated which are also a problem. In today‘s community in which time plays an important role, multi document summarizer play essential role in such situations. So, recent year most researchers for automatic text summarization have transferred their efforts from single documents to multiple documents. So this section, only deal to compare technique for extraction summarization from multi document. There are many techniques that people use for multi document summarization from the past to present. Multi document summarization became more interested by the mid 1990s, Summary of multi document must include the important ideas in each document, comparing ideas across the document, reducing the size of each document and ordering in a new sentence. The first started of multi document by Radev and McKeown (1995) presented in [17] developed SUMMONS to generate summaries of multiple documents on the same or related events, presenting similarities and differences, contradictions, and generalizations among sources of information from realized as English sentences. In 1998 [18], they improved their SUMMONS to combine it into a conceptual representation of the summary which selects information from underlying knowledge base. The structured conceptual representation of the summary, where information that appears in only one article is given a lower rating and information that is synthesized from multiple articles is rated highly.

The paper [19] discusses a text extraction approach to multi document summarization that builds on single document summarization methods by using additional, available information about the document set as a whole and the relationships between the documents. Multi document summarization differs from single in that the issues of compression, speed, redundancy and passage selection are critical in the formation of useful summaries. This approach addresses these issues by using domain independent techniques based mainly on fast, statistical processing, a metric for reducing redundancy and maximizing diversity in the selected passages, and a modular framework to allow easy parameterization for different genres, corpora characteristics and user requirements.

The paper [20] proposed a summarization system which automatically classified type of the document set and summarized a document set with its appropriate summarization mechanism. This system classified a document set into three types, a series of events, a set of the same events and related events, by using information of higher frequency nouns and named entity.

The unnecessary parts are deleted after summarizing each document and generated multi document summary. They used single document summarization mechanism for each document of a document set and removed similar parts between summarized documents for generation of a target summary. They applied a TF/IDF based sentence extraction for single document summarization and used of single document summarization for multi document summarization. Their mechanism of document set classification does not work well in the evaluation because their current implementation has some system bugs in classification mechanism.

In recent trend, some work of multi document summarization extends to multi language environment which proposed by Evans et al., in 2005 presented in [21].

2.3 Summarization in Nepali Language

No automated Text Summarizer for single document as well as for multi documents are yet discovered for Nepali text documents. There are a few researches in the field of natural language processing done so far. Most of such researches rally on lab and includes limited area.

Chapter 3

RESEARCH METHODOLOGY

This chapter describes the theoretical concept behind the methods used in this dissertation work. Model of the proposed text summarization system is given in section 3.1. Research methodologies describe the stepwise solutions to the problem of various summary generation steps.

3.1 System Overview

The top level system model diagram of the proposed system is given in Figure 3.1. Various stages have to be performed to achieve automatic text summarization for the Nepali documents. Detailed sub-system flow is given in Figure 3.2.

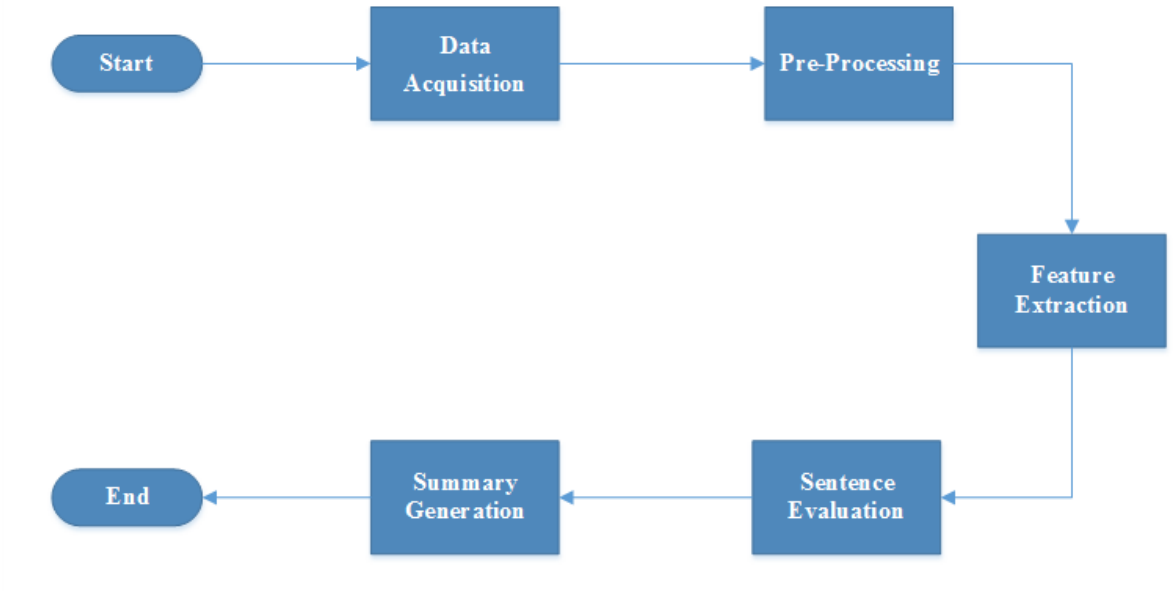

Figure 3.1: Top Level System Model.

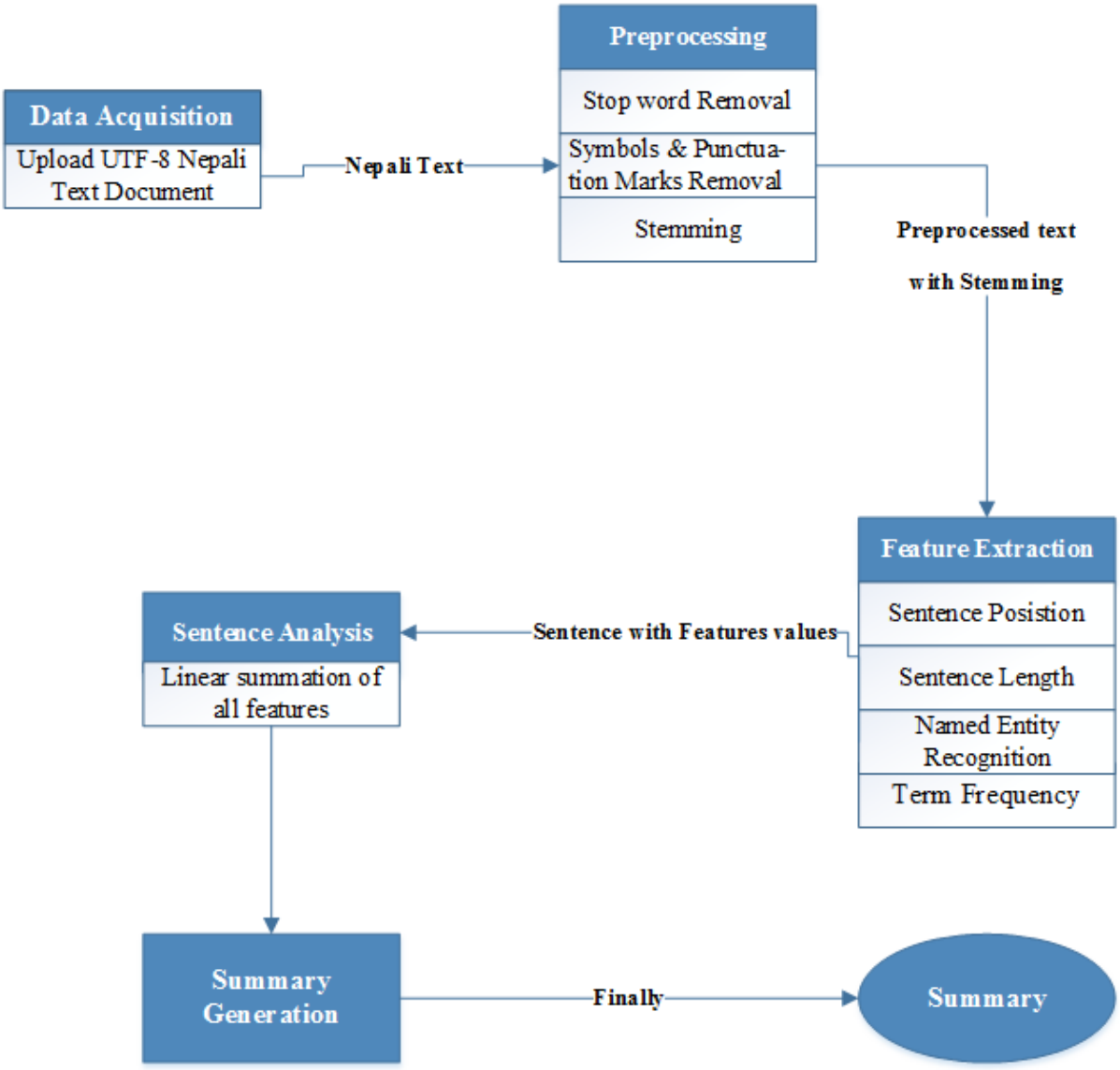

Figure 3.2: Detail Architecture of System Model.

3.2 Data Acquisition

This step is considered to be the system input. The system input is a user text input. In the user text input, the user is asked to provide a UTF-8 Unicode Nepali text either from user interface or should give the text file with (.txt) format. This Nepali text or file is one to be summarized. There is no limitation on the size of the text or number of sentences included in the document. Not only that, there is no restriction on the domain of documents. So it may be related to either sports, politics, medical or other.

3.3 Preprocessing

The preprocessing stage is just for removing the words which do not carry valueable information. So, in this stage, we need to remove any extraneous symbols, punctuation marks and stop

words from each sentence of the document. Then after, we need to perform stemming of each word in each sentence available in the document to make a sentence having less features.

3.3.1 Symbols and Punctuation Marks Removal

There will be some symbols included in the document to represent some information. They are not so informative, like the symbols $, #, , %, etc are used to denote some information in the document. So, we need to remove such symbols from the document before feature extraction. As in any language, punctuation marks are used to organize the text and to make meaningful sentences. The punctuation in the text summary does not have any value, so we remove all punctuation which are not full stop. The data dictionary used for symbol removal is explained in section 5.2.

3.3.2 Stop Words Removal

Stop words are high frequency words of a language which rarely contribute to useful information in terms of document relevance and appear frequently in the text but provide less meaning in identifying the important content of the document [12]. Closed class words such as pronoun, prepositions, conjunctions, etc available in the document are often included in stop words list. Stop words are pruned at the processing phase to reduce the number of features. The stop word lists for English and other languages are freely available on the Web and often utilized in summarization. But, we cannot find easily the stop word list for Nepali language. We have prepared the list of stop words for Nepali language manually for this dissertation. During the removal procedure all the words that appear in a list of stop words are removed by matching from the source documents. Some of the Nepali stop words are given in Figure 3.3. The details of dictionary used for stop word removal is explained in Section 5.2

छ, म, हो, छु, केही, कोही, हामी, मेरो, त्यो, को, हरु, फेरी, हाम्रो, अर्को

Figure 3.3: Some List of Nepali Stop Words.

3.3.3 Stemming

Stemming is an essential process in the field of NLP. Word stemming is the process of reducing inflected or derived words to their stem, base or root form. Many NLP applications which use words as basic elements employ stemmers to extract the stems of words. Mainly, it is used in information retrieval systems to improve performance. Actually, this operation reduces the number of terms in the information retrieval system, thus decreasing the size of the index files. Stemming helps to obtain the stem or root of each word, which ultimately helps in semantic analysis and faster processing. There is a need of specific language dependent stemmer, and is requires some significant linguistic expertise in the language. A typical simple stemmer

algorithm involves removing suffixes/prefixes using a list of frequent suffixes/prefixes, while a more complex one would use morphological knowledge to derive a stem from the words. The stemmer which simply prunes the suffixes/prefixes using the list of frequent suffixes/prefixed is very efficient and lightweight approach compared to morphological parsing. Even though, there are some advanced stemmers for languages such as English, the algorithms which they employ do not work well for highly inflected languages such as Nepali

Since Nepali is a highly inflected language so there are many word forms to denote a single concept. This situation is highly effected for the frequency of a term and therefore words have to be stemmed out before getting their frequencies. There is light weight stemmer for Nepali language [22] and we incorporate this stemmer in our system to get the root word of the inflected words.

3.4 Feature Extraction

After removing the unnecessary words, symbols, prefixes and suffixes of the terms/words from each sentence of the text document, we need to evaluate the features of each sentence that will play vital role in the sentence evaluation. In feature extraction, we need to calculate the various features of each sentence in the document like length feature, position feature, entity feature, frequency feature, etc. The details of finding the various features of sentence are described in subsequent sections.

3.4.1 Length of Sentence

This feature is useful to filter out too short or too long sentences such as subtitles, author names and date lines commonly found in the articles. When such short and long sentences do not carry the meaningful information, they are not needed to be included in the summary. We calculate this feature relative to longest sentence available in the document using the following equation [12]:

$$Sentence\ relative\ length = \frac{Number\ of\ words\ in\ sentence}{Number\ of\ words\ in\ longest\ sentence} \tag{3.1}$$

For example: if we have 45 words in longest sentence of the document and one of the sentence contains only 10 words after pre-processing stage, then the sentence relative length for that sentence in the document will be $= 10/45 = 1/4.5$.

3.4.2 Position of Sentence

Position of sentence in the document plays an important role in finding the sentence that is the most relevant to the topic of the document. Usually the first sentence is the most significant in the document which gives an idea about what the rest of the text say, and the last sentence usually contains a conclusion about the text.

To stress the significance of different sentence positions, each sentence in the document is given a rank ranging from 1 to some max value. More weight is given to sentence at the beginning than the rest. We compute sentence absolute position feature using the equation [12]:

$$Sentence\ absolute\ position = \frac{Number\ of\ sentences\ -\ sentence\ position\ +\ 1}{Number\ of\ Sentences} \quad (3.2)$$

For example: if we have a document with 11 sentences, then the sentence absolute position for the first sentence in the document will be $= (11 - 1 + 1)/11 = 11/11 = 1$.

3.4.3 Named Entity Recognition

The motivation for this feature is that the occurrence of proper nouns, referring to people, places and other categories, are clues that a sentence is relevant to the summary.

For each sentence of a document, we check if there is any entity name occurs in it against the dictionary of NE, if it occurs, we increase a entity counter variable by 1 and remove this entity name from a pre-process-sentence. The value of this counter reflects the number of entities in the sentences. It also plays an import role in the sentence analysis stage to determine which sentence is to be included in the summary. If the documents contains the more entities then we can say the documnent is about to these entities and the summary should include sentence having these entites.

This stage calculates the feature for following four entities:

- Person Name
- Location Name
- Organization Name
- Miscellaneous Named Entities

3.4.4 Term Frequency

Term frequency is a numerical statistic that reflects how important a word is to a document. It is used as a weighting factor in information retrieval and text mining [23]. The term frequency value increases proportionally to the number of times a word appears in the document which helps to recognize the commonness and generality of any word in the document.

In automatic text summarization, we have to select a set of relevant sentences to be included in the extractive summary out of all sentences in a document. Hence, the notion of a collection of document in information retrieval can be replaced by the notion of a single document in

text summarization. This new measure will be called Term Frequency x Inverse Sentence Frequency (TF-ISF) [24]. We use the following formula to calculate TF-ISF:

$$TF\ X\ ISF = \frac{\sum_{i=1}^{k} w_i(S)}{\max \sum_{i=1}^{k} w_i(S^N)} \tag{3.3}$$

Where k is the number of words in a sentence (S), w is the term frequency weight for each word i and N is the total number of sentences in a document.

3.5 Sentence Evaluation

The sentences are evaluated based on the score they have and it is computed using the linear combination of the normalized values of weighted features collected in Section 3.4. Equation 3.4 shows the score of a particular sentence.

$$\begin{aligned} Score(s) = Sentence\ Relative\ Length + Sentence\ Absolute\ Position + Person \\ Name\ Counter + Location\ Name\ Counter + Organization \\ Name\ Counter + Misc\ Named\ Entity\ Counter + TF\ X\ ISF \end{aligned} \tag{3.4}$$

From the above equation, we can determine the final weight of each sentence in the test corpus. Based on these numric values, the sentences are determined whether they are included in the summary or not.

3.6 Summary Generation

All document sentences are sorted in descending order of their scores assigned in sentence evaluation step. A set of highest score sentences are extracted as document summary. Finally the summary sentences are arranged in the original order to ensure the readability of the generated summary. The number of sentences to be included in the summary will be depended on the number of sentences in the document. For all the document, the default summary will include $\frac{1}{3}rd$ sentence of the original document, but we can generate summary of user given percentage.

3.7 System Evaluation Metrics

3.7.1 Precision/Recall/F-Score

System evaluation is a hard task because it is very difficult to find an ideal text summary for a given document or a set of documents. Another problem is that two manual summaries of the same input do not in general share many identical sentences. To evaluate our system, we

used human generated reference summary for each document in the collection. Then for each reference summary and document, we will calculate the three important measures: precision, recall and F-measure [12]. This type of evaluation technique is called Evaluation by Sentence Co-selection.

Precision is a measure of how many of the sentences generated by the system is correct and it is calculated as:

$$Precision = \frac{Number\ of\ system\ correct\ summmary\ sentences}{Number\ of\ system\ summary\ sentences} \tag{3.5}$$

Recall is a measure of how many of the sentences in the reference summary that are present in the system generated summary and it is calculated as:

$$Recall = \frac{Number\ of\ system\ correct\ summmary\ sentences}{Number\ of\ human\ summary\ sentences} \tag{3.6}$$

Usually Recall and Precision are opposite to one another. A system strives for coverage will get lower precision and a system strives for precision will get lower recall. F-measure balances recall and precision using a parameter β. The F-measure is calculated as [12]:

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{3.7}$$

When β is one, Precision P and Recall R are given equal weight. When β is greater than one, Precision is favored, when β is less than one, recall is favored. In the following experiments β equals one.

3.7.2 Cosine Similarity

We can clear out the above discussed weakness of co-selection measures by content-based similarity measures. The content˘based similarity measures is obtained from Cosine similarity [25].

Cosine similarity is a measure of similarity between two document vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0^0 is 1, and it is less than 1 for any other angle. It is thus a judgement of orientation and not magnitude: two vectors with the same orientation have a Cosine similarity of 1, two vectors at 90^0 have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in $[0, 1]$.

The cosine of two vectors can be derived by using the Euclidean dot product formula:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta \tag{3.8}$$

Given two vectors of attributes, A and B, the cosine similarity, $cos(\theta)$, is represented using a dot product and magnitude as

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{3.9}$$

The resulting similarity ranges from -1 meaning exactly opposite, to 1 meaning exactly the same, with 0 usually indicating independence, and in between values indicating intermediate similarity or dissimilarity. For text matching, the attribute vectors A and B are usually the term frequency vectors of the documents. The cosine similarity can be seen as a method of normalizing document length during comparison. In the case of information retrieval, the cosine similarity of two documents will range from 0 to 1, since the term frequencies ($tf - idf$ weights) cannot be negative. The angle between two term frequency vectors cannot be greater than 90^0.

Chapter 4

IMPLEMENTATION

All the modules are implemented by using Java (spring MVC/JSP) JDK 1.7 version. The Java is installed on a Intel(R) Core(TM)2 Duo CPU T6500 @ 2.10GHz processor. The Computer has total main memory of 2 Gigabyte and 32-bit Microsoft Windows 7 Professional operating system installed in it. This chppter describes the algorithm of some major stages used in the system implementation along with their respective output of some input sample.

4.1 Document Preprocessing Algorithms

4.1.1 Symbols and Punctuation Marks Removal Algorithm

The data dictionary of symbol and puntuation mark used in this work is presented in Section 5.2. Such symbol marks presented in document are removed using following algorithm.

Algorithm 4.1 Symbols and Punctuation marks Removal

1: Read text document.
2: Match the token of document with token in the symbol and punctuation mark dictionary.
3: Remove matched token from document.
4: Repeat until all the symbols and puntuation marks are not removed from the document.
5: Stop.

4.1.2 Stop Words Removal Algorithm

The data dictionary used for stop words is given in Section 5.2 are removed from the input document using the algorithm below.

Algorithm 4.2 Stop Word Removal

1: Read text document.
2: Match the token of document with token in the stop word dictionary.
3: Remove matched token from document.
4: Repeat until all stop words are not removed from the document.
5: Stop.

4.1.3 Stemming Algorithm

To obtain the root word of the inflected words presented in the documents, we use the following algorithm [22].

Algorithm 4.3 Stemming

1: Read text from input document.
2: Do the following for the string sequence in the input word
3: Strip off ेी appeared at the end of the word, the very last letter of the input word.
4: Strip off ेीय/ेीया appeared at the end of the word from the end of the input word.
5: Strip off ेाइ from the input word which is appeared at the end of the word from the end. Add े to the end of the resulting word if the last letter of the word formed is a consonant.
6: Exception holds the letter व. If the last character of the resulting word is व, strip it off and add ेा .
7: If the initial letter is a vowel, stripe off ेाइ from the word and insert ेा in front of the character which is followed by ेाइ .
8: Stripe off े from the end of the word. Look for the resulting word in the free morpheme list.
9: The remaining part of the input word is root word.
10: Stop.

4.2 Summary Generation Steps Demonstration

The steps of system symmary generation explained in chapter 3 are demonstrated in this section. For the demonstration purpose, one document from the corpus is taken as sample which is shown in figure 4.1. The input sample contains 8 sentences and the system summary for this input have only 3 senetences. The mannual summary of this input includes 4 sentences.

The output of each stage for the sample inut are provided below.

4.2.1 Input Document

The sample input is;

Input:	
S.N.	**Sentences**
1	अपाङ्गहरु हाम्रै बन्धुबान्धव हुन् र यसै देशका नागरिकहरु हुन् ।
2	उनीहरुमा पनि अरुजस्तै इच्छा, आकान्क्षा, भावना वा चाहनाहरु हुन्छन् ।
3	हामीले उनीहरुलाई छि : छि : र दूरदूर गर्ने होइन सम्मानित र मर्यादित जीवन बाच्ने अवसर दिनुपर्छ ।
4	अपाङ्ग हुनेबित्तिकै उसका सोचाई, कल्पनाशीलता र कर्यक्षमता पनि अपाङ्ग हुन्छन् भन्ने सोच्नु मूर्खता बाहेक केही होइन ।
5	उपयुक्त अवसर प्राप्त भए उनीहरुले पनि उल्लेखनीय कार्य गर्न सक्दछन् ।
6	हेलेन केलेर, जोन मिल्टन जस्ता व्यक्तित्वहरु अपाङ्ग भए पनि आफ्ना बेजोड प्रतिवाले गर्दा संसारमै प्रसिद्द छन् ।
7	नेपालका इतिहास शिरोमणि बाबुराम आचार्यले आँखाको ज्योति गुमाएर पनि इतिहासको अध्ययन र लेखनकार्यमा आजीवन विश्राम लिएनन् ।
8	यसरी अपाङ्ग नागरिकले पनि प्रतिभा, ज्ञान र सीपका माध्ययमबाट महत्त्वपूर्ण योगदान पुरयाउन सक्दछन् ।

Figure 4.1: Sample of Input Document.

4.2.2 Symbosl and Punctuation Marks Removal

The output after removing the symbols and punctuation marks is obtained as;

Symbol and Punctuation marks Removal	
S.N.	**Sentences**
1	अपाङ्गहरु हाम्रै बन्धुबान्धव हुन् र यसै देशका नागरिकहरु हुन्
2	उनीहरुमा पनि अरुजस्तै इच्छा आकान्क्षा भावना वा चाहनाहरु हुन्छन्
3	हामीले उनीहरुलाई छि छि र दूरदूर गर्ने होइन सम्मानित र मर्यादित जीवन बाच्ने अवसर दिनुपर्छ
4	अपाङ्ग हुनेबित्तिकै उसका सोचाई कल्पनाशीलता र कर्यक्षमता पनि अपाङ्ग हुन्छन् भन्ने सोच्नु मूर्खता बाहेक केही होइन
5	उपयुक्त अवसर प्राप्त भए उनीहरुले पनि उल्लेखनीय कार्य गर्न सक्दछन्
6	हेलेन केलेर जोन मिल्टन जस्ता व्यक्तित्वहरु अपाङ्ग भए पनि आफ्ना बेजोड प्रतिवाले गर्दा संसारमै प्रसिद्द छन्
7	नेपालका इतिहास शिरोमणि बाबुराम आचार्यले आँखाको ज्योति गुमाएर पनि इतिहासको अध्ययन र लेखनकार्यमा आजीवन विश्राम लिएनन्
8	यसरी अपाङ्ग नागरिकले पनि प्रतिभा ज्ञान र सीपका माध्ययमबाट महत्त्वपूर्ण योगदान पुरयाउन सक्दछन्

Figure 4.2: Symbols and Punctuation Marks Removed Output.

4.2.3 Stop Word Removal

After removing the stop words, the output is obtained as;

Stopword Removal	
S.N.	**Sentences**
1	अपाङ्गहरु बन्धुबान्धव देशका नागरिकहरु
2	इच्छा आकान्क्षा भावना चाहनाहरु
3	दूरदूर सम्मानित मर्यादित जीवन बाच्ने अवसर दिनुपर्छ
4	अपाङ्ग हुनेबित्तिकै सोचाई कल्पनाशीलता कर्यक्षमता अपाङ्ग सोच्नु मुर्खता
5	उपयुक्त अवसर प्राप्त उल्लेखनीय
6	हेलेन केलेर जोन मिल्टन व्यक्तित्वहरु अपाङ्ग बेजोड प्रतिवाले संसारमै प्रसिद्द
7	नेपालका इतिहास शिरोमणि बाबुराम आचार्यले आँखाको ज्योति गुमाएर इतिहासको अध्ययन लेखनकार्यमा आजीवन विश्राम
8	अपाङ्ग नागरिकले प्रतिभा ज्ञान सीपका माध्ययमबाट महत्त्वपूर्ण योगदान पुरयाउन

Figure 4.3: Stop Word Removed Output.

4.2.4 Stemming

The stemming output is;

Stemming	
S.N.	**Sentences**
1	अपाङ्गहरु बन्धुबान्धव देश नागरिकहरु
2	इच्छा आकान्क्षा भाव चाहनाहरु
3	दूरदूर सम्मानित मर्यादित जीवन बाच्ने अवसर दि
4	अपाङ्ग हुनेबित्तिकै सोच कल्पना कर्यक्षमता अपाङ्ग सोच मुर्खता
5	उपयुक्त अवसर प्राप्त उल्लेखनीय
6	हे केलेर जो मिल्टन व्यक्तित्वहरु अपाङ्ग बेजोड प्रतिवा संसार प्रसिद्द
7	नेपाल इतिहास शिरोमणि बाबुराम आचार्य आँखा ज्योति गुमाएर इतिहास अध्ययन लेख आजीवन विश्राम
8	अपाङ्ग नाग प्रतिभा ज्ञान सीप माध्ययमबाट महत्त्व योग पुरयाउन

Figure 4.4: Stemming Output.

4.2.5 System Summary

The system summary for the above input sample is obtained as;

System Summary	
S.N.	**Sentences**
1	अपाङ्ग हुनेबित्तिकै उसका सोचाई, कल्पनाशीलता र कर्यक्षमता पनि अपाङ्ग हुन्छन् भन्ने सोच्नु मूर्खता बाहेक केही होइन ।
2	हेलेन केलेर, जोन मिल्टन जस्ता व्यक्तित्वहरु अपाङ्ग भए पनि आफ्ना बेजोड प्रतिवाले गर्दा संसारमै प्रसिद्द छन् ।
3	नेपालका इतिहास शिरोमणि बाबुराम आचार्यले आँखाको ज्योति गुमाएर पनि इतिहासको अध्ययन र लेखनकार्यमा आजीवन विश्राम लिएनन् ।

Figure 4.5: System Summary Output.

4.2.6 Mannual Summary

The mannual summary for the above input sample is given below. The process of obtaining the manual summary is described in section 5.1.

Mannual Summary	
S.N.	**Sentences**
1	अपाङ्ग हुनेबित्तिकै उसका सोचाई, कल्पनाशीलता र कर्यक्षमता पनि अपाङ्ग हुन्छन् भन्ने सोच्नु मूर्खता बाहेक केही होइन ।
2	उपयुक्त अवसर प्राप्त भए उनीहरुले पनि उल्लेखनीय कार्य गर्न सक्दछन् ।
3	हेलेन केलेर, जोन मिल्टन जस्ता व्यक्तित्वहरु अपाङ्ग भए पनि आफ्ना बेजोड प्रतिवाले गर्दा संसारमै प्रसिद्द छन् ।
4	नेपालका इतिहास शिरोमणि बाबुराम आचार्यले आखाको ज्योति गुमाएर पनि इतिहासको अध्ययन र लेखनकार्यमा आजीवन विश्राम लिएनन् ।

Figure 4.6: Manual Summary.

4.3 Document Feature Vector and Sentence Ranking

Table 4.1 shows the sentence features obtained during summary generation for the 8 sentences of the input sample document given in Figure 4.1. It also contains the sentence score of each sentence with their rank value.

Table 4.1: Document Features and Sentence Ranking.

S.N.	Length Feature	Position Feature	Person NE Feature	Location NE Feature	Organization NE Feature	Misc. NE Feature	TS-ISF Feature	Sentence Score	Sentence Rank
1	0.50	1.00	0	0	0	0	18.71	20.21	6
2	0.50	0.88	1	0	0	0	14.44	16.81	7
3	0.86	0.75	1	0	0	0	24.26	26.87	4
4	0.86	0.63	1	0	0	0	31.07	33.56	2
5	0.43	0.50	0	0	0	0	13.86	14.79	8
6	0.86	0.38	0	0	0	1	24.95	27.19	3
7	1.00	0.25	3	1	0	0	30.97	36.22	1
8	0.71	0.13	1	0	0	1	21.49	24.33	5

Chapter 5

EXPERIMENTATIONS AND RESULTS

This chapter contains all the datasets and data dictionary used in the experiments and corresponding empirical results. There are five self created datasets used for the evaluation of the system. The datasets and data dictionary are given in Section 5.1 and 5.2 respectively . Experimentation results and graphical analysis are provided in Section 5.3.

5.1 Testing Datasets

For testing purpose, we collected total 38 documents from various domains. Among them, 10 documents are selected randomly from books and 28 documents are selected randomly from national news papers of Nepal in the time period of June 2013 through December 2013. The number of sentence containing in each document are varied. Detail of each dataset is given below.

We have created the human reference summary for the above collected document with the help of Nepali expert (Lecturer in Higher Secondary School). The expert was guided with some pre-assumption during mannual summary generation of those documents. Actually, the pre-assumption includes following points:

- The number of sentences to be included in the human reference/mannual summary should be 1/3rd of total number of sentence in the documents as far as possible.
- The reference summary should be extracts not the abstracts. That is, the sentence to be extracted for the reference summary should be exactly same of the document. Since, we need the extract summary to validate the system.

5.1.1 Dataset 1: Book

Dataset 1 contains 10 different documents collected from different Nepali books of Secondary and Higher Secondary Level. Documents belongs to variety of categories like sports, life, science, nature, etc.

5.1.2 Dataset 2: Kantipur News

Dataset 2 contains 6 different documents of different domains including political news, sports, etc. which are collected from Kantipur National daily news paper of Nepal.

5.1.3 Dataset 3: Himalkhabar Patrika

Dataset 3 contains 4 different documents collected from Himal Khabar Patrika, the monthly news paper of Nepal.

5.1.4 Dataset 4: Nagarik News

Dataset 4 contains 6 different documents of various domains collected from Nagrik news, the national daily news paper of Nepal.

5.1.5 Dataset 5: Online Khabar Patrika

Dataset 5 contains 12 different documents of diffent domains collected from Online Khabar Patriks, the national daily news paper of Nepal.

5.2 Data Dictionaries

5.2.1 Symbols and Punctuation Marks Dictionary

The symbols that doesn't carry the special meanings are removed in the pre-processing stage. The symbol and punctuation marks dictionary contains 40 unique symbols and punctuation marks. This dictionary includes the following symbols and marks.

!	,	:	'	÷	×	º	><	|	—
¿	)	\	@	#	$	%	^	*	،
(	_	-	+	=	~	ø	"	[	]
‘	’	/	|	”	&	:-	“	;	?

Figure 5.1: Symbol and Punctuation Marks Dictionary.

5.2.2 Stop Word Dictionary

We have created the stop word dictionary to remove the useless words from the document in pre-processing stage. The words in the document that matched with the words listed in the stop word dictionary are excluded. The dictionary of stop word contains 402 unique words. The stop word dictionary contains following words:

अँ	अगि	अचेल	अझ	अझै	अति	अत्यन्तै	अथवा	अनुसार	अब	अरु
अरुजस्तै	अरे	अनि	अर्का	अर्काको	अर्कै	अर्को	अलि	अलिक	अलिकति	अहिले
आ	आइ	आए	आज	आजै	आत्था	आफू	आफ्ना	आफ्नै	आफ्नो	आम्मै
आयो	आहा	उक्त	उठे	उता	उति	उनका	उनको	उनी	उसका	उसको
उसो	उहां	ए	एक	एक्लै	एवं	ऐया	ओ	ओहो	औंधी	कठै
कता	कति	कम	कसरी	कसो	कहाँ	का	कार्य	कि	किन	किनभने
की	कुनै	कुन्नि	कुरा	के	केही	कै	कैले	को	कोही	क्यारे
क्यावात्	क्रम	क्वाप्प	खुब	खुसुक	खूरुखूर	खै	गई	गए	गएको	गजक्क
गत	गयो	गराई	गरियो	गरी	गरे	गरँ	गर्दा	गर्दै	गर्न	गर्नू
गर्ने	गर्नेछ	घटे	चरक्क	चले	चाँही	चासो	चिटिक्क	छ	छन्	छि
छिटो	छु	छुटे	छैन	छौ	जता	जति	जब	जय	जस	जसको
जसरी	जस्ता	जस्तै	जहाँ	जहिले	जुन	जे	जैले	जो	जोड	ज्या
ज्यादै	झनक्क	झमक्क	झलल	टहटह	ठुलो	ढकमक्क	ढङ्ग	तँ	तथा	तथापि
तथ्य	तर	तर्फ	तल	तसर्थ	तह	तापनि	तिनले	तिनी	तिमीले	तिर
तुरुक्क	तुरुन्तै	तैपनि	त्यता	त्यति	त्यस	त्यसरी	त्यसै	त्यस्तै	त्यहाँ	त्यहीं
त्यो	थप	थपक्क	थिए	थोरै	थ्याच्च	दाँया	दिई	दिए	दिनु	दिने
देखि	द्वारा	धत्	धन्य	धपक्क	धिक्कार	धेरै	न	नगरे	नजिक	नभई
नभए	नयाँ	नि	निकै	निम्ति	नै	पक्ष	पछि	पढ्न	पनि	पर
पर्छ	पर्दछ	पर्ने	पर्सि	पाए	पारि	पार्नु	पिलपिल	पुलुक्क	पो	प्याच्च
प्रति	प्रशस्त	फतफत	फेरि	बने	बन्नु	बमोजिम	बहुत	बाँया	बाट	बारे
बाहिर	बाहेक	बिचरा	बिहान	बीच	बुझे	बुझ्यौ	बुलुक्क	बेलुका	बेसरी	बोले
ब्यारे	भई	भए	भनिए	भने	भने	भनेर	भन्नु	भन्ने	भयो	भर
भरे	भर्खर	भित्र	भुल्भुल	भै	भो	भोलि	म	मा	मात्र	माथि
मुनि	मुसुक्क	मेरो	मै	मैले	यता	यति	यस	यसको	यसमा	यसरी
यसर्थ	यसलाई	यसै	यसो	यस्ता	यस्तै	यस्तो	यहाँ	यहि	या	यी
यो	र	रहयो	रहे	राख	राखे	रामरी	रूप	रे	लगे	लाई
लागि	लिई	लिए	लिन	लिनु	लिने	ले	लेख्न	लौ	लौन	वर
वा	वाट	वाटै	वारि	वारेमा	वाह	श्री	सँग	संघ	सक्छ	सक्नु
सधै	सबैरै	सबै	समेत	सम्म	सरर	साथ	साथै	सानो	सार्है	सिमसिम
सुटुक्क	सो	सोही	स्याबास	हँ	हगि	हजुर	हने	हरु	हरेक	हामी
हामीले	हामीहरुको	हाम्रै	हाम्रो	हाल	हिजो	हुँदा	हुँदै	हुँदैन	हुना	हुनु
हुने	हुन्	हुन्छ	हुन्छन	हे	हेरे	हेर्न	है	हो	होइन	होला
होस्	पर्दैन	भन्दा	हेरेर	सिधै	झन्झन्	बन्दै	आएको	लागेको	भएकाले	त
यसका	कसैलाई	त्यसैले	गर्नुका	त्यसको	आजको	हालको	क्रममा	आफूले	तोकेको	भन्नेवारे
नपाउने	गर्नुपर्ने	गर्नुपर्छ	कतिपय	भन्नेतर्फ	सवभन्दा	रुपमा	रहेका	आफैले	बन्न	सकिन्छ
कुरालाई	राख्दै	बनाउदै	बनाउन	खुला	कसैको	नहुने	अन्य	आदि	ईत्यादी	यसअघि
अघि	गरिने	माझ	मान्य	रुप	रहने					

Figure 5.2: Stop Word Dictionary.

5.2.3 NE Dictionary

To test ATSSN, we have to prepare the data dictionary for NE (Named Entity). The NE corpus that is used for testing ATSSN is taken from [26] with some addition of corpus, which was created manually and contains $21,105$ unique words. The detail description of NE corpus is

shown in below table.

Table 5.1: NE Dictionary

Dictionary	**No. of entries**
Person Name	5478
Location Name	5243
Organization Name	4715
Miscellaneous Name	5669
Total	**21105**

5.3 Experimentation Results

To test the system, we have to compare the system generated summary with human reference summary. With the help of Precision (P), Recall (R), F-Score (F) and Similarity of system summary with reference summary, we calculated the accuracy of the system. The experimentation results obtained from different experiments are explained in this section.

For the experimentation, we categorize the testing document into three different sets. The first set contains the decuments having the number of sentences less than 15. Similarly, the second set includes the documents having the number of sentece greater than 15 and less than or equal to 30. And, the third set contains any number of sentence more than 30.

5.3.1 Experiment 1

This experiment is carried out in 14 different documents from datasets described in Section 5.1. In this experiment, each document contains number of sentences upto 15 and for the docunents having same number of sentence, the average value of each measures parameter is calculated and shown in table 5.2. This experiment shows that the average Precision, Recall and F-Score are obtained as 75.72% ,64.05% and 68.99% respectively. And, the average cosine similarity between manual and system summary is obtained as 84.89%. Figure 5.3 shows the graph between Precision, Recall and F-Score measures with respect to number of sentences in the documents,manual summary and system summary. And, Figure 5.4 shows the graph of manual and system summary similarity with respect to number of sentences in the documents,manual summary and system summary.

Table 5.2: Result of Experiment 1.

No. of Docs	Avg. NDS	Avg. NMSS	Avg. NSSS	Avg. MSP	Avg. SSP	Precission (%)	Recall (%)	F-score (%)	Similarity (%)
4	8	4	3	50.00	37.50	75.00	56.25	64.28	82.22
1	9	3	3	33.33	33.33	66.67	66.67	66.67	81.58
3	10	4	3	40.00	30.00	77.78	58.33	66.66	85.25
4	13	4.5	4	34.61	30.76	75.00	67.50	70.50	85.36
1	14	5	5	35.71	35.71	80.00	80.00	80.00	90.00
1	15	5	5	33.33	33.33	80.00	80.00	80.00	90.84

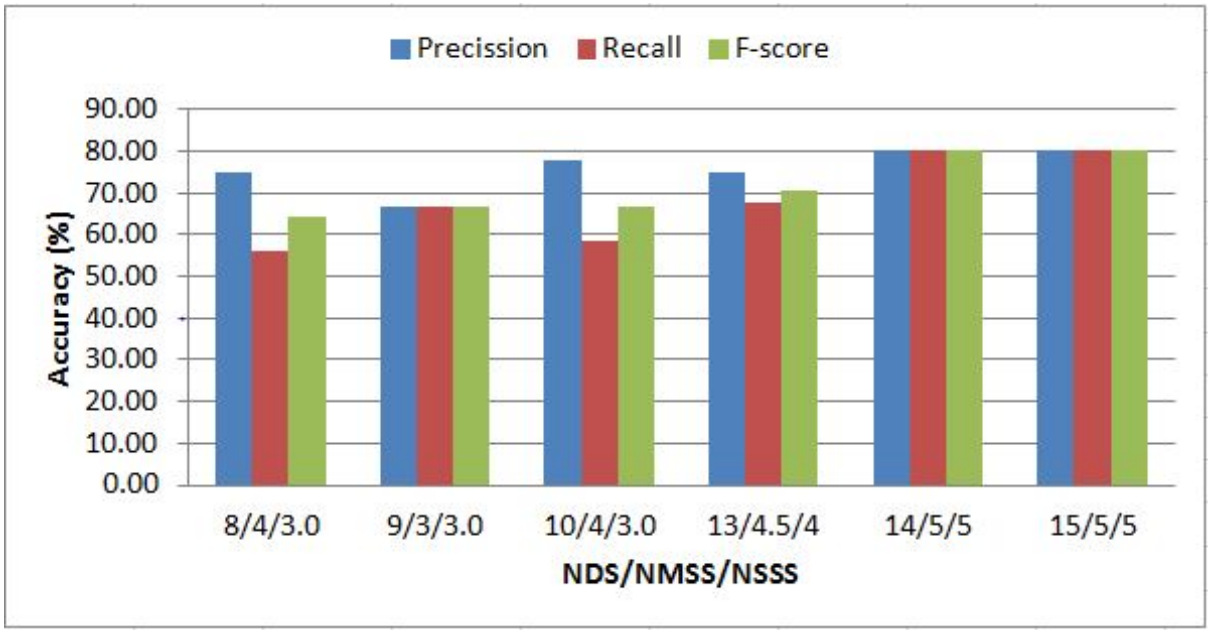

Figure 5.3: Precision-Recall-FScore Graph of Experiment 1.

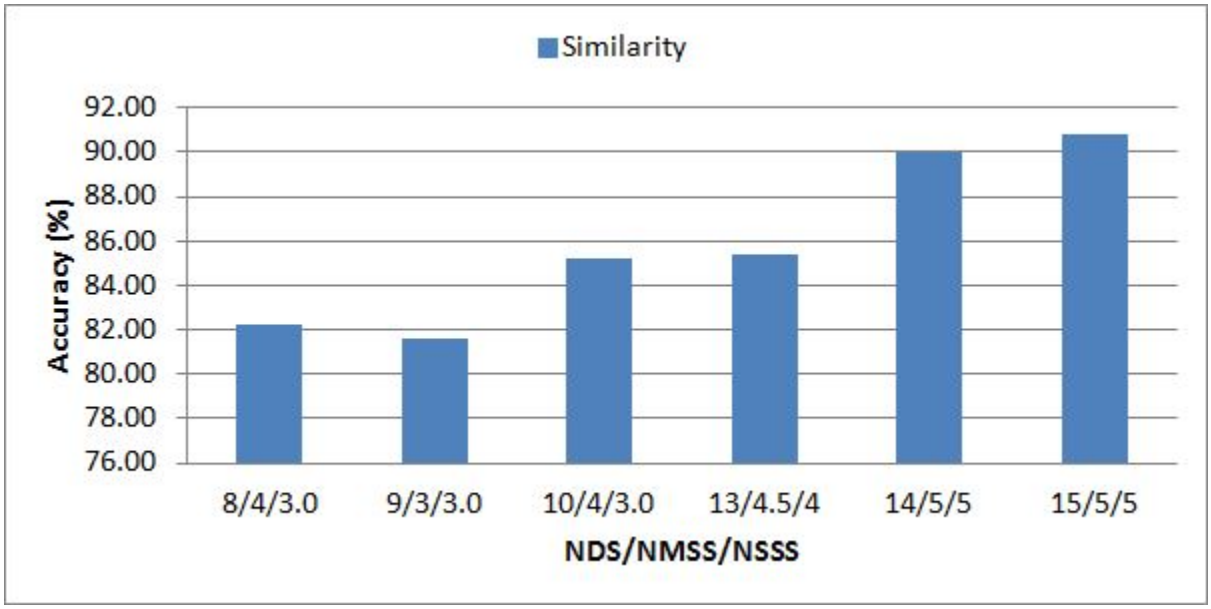

Figure 5.4: Similarity Graph of Experiment 1.

5.3.2 Experiment 2

This experiment is carried out in 12 different documents from datasets described in Section 5.1. In this experiment, each document contains number of sentences between 15 and 30. For the docunents having same number of sentence, the average value of each measures parameter is calculated and shown in table 5.3. This experiment shows that the average Precision, Recall and F-Score are obtained as 79.74% ,73.06% and 76.11% respectively. And, the average cosine similarity between manual and system summary is obtained as 92.54%. Figure 5.5 shows the graph between Precision, Recall and F-Score measures with respect to number of sentences in the documents,manual summary and system summary. And, Figure 5.6 shows the graph of manual and system summary similarity with respect to number of sentences in the documents,manual summary and system summary.

Table 5.3: Result of Experiment 2.

No. of Docs	Avg. NDS	Avg. NMSS	Avg. NSSS	Avg. MSP	Avg. SSP	Precission (%)	Recall (%)	F-score (%)	Similarity (%)
2	16	5.5	5	34.375	31.25	80	73.335	76.365	89.485
3	17	6.67	5.67	39.95	33.94	82.22	69.84	75.52	89.86
1	19	7	6	36.84	31.57	66.67	57.14	61.54	98.94
1	20	8	7	40	35	85.71	75	80	90.5
1	24	9	8	37.5	33.33	87.5	77.78	82.35	93.29
1	25	7	8	28	32	62.5	71	66.67	95.93
1	28	9	9	32.14	32.14	77.78	77.78	77.78	90.93
1	29	11	10	37.93	34.48	90	81.82	85.71	99.22
1	30	10	10	33.33	33.33	80	80	80	93.14

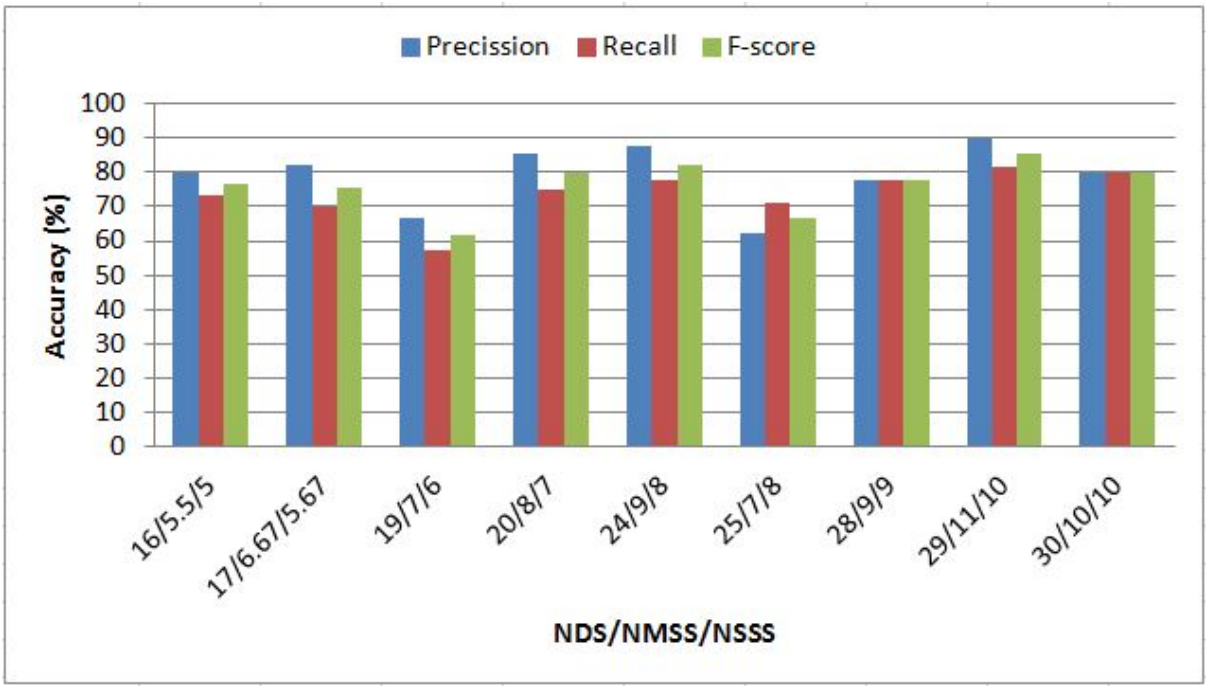

Figure 5.5: Precision-Recall-FScore Graph of Experiment 2.

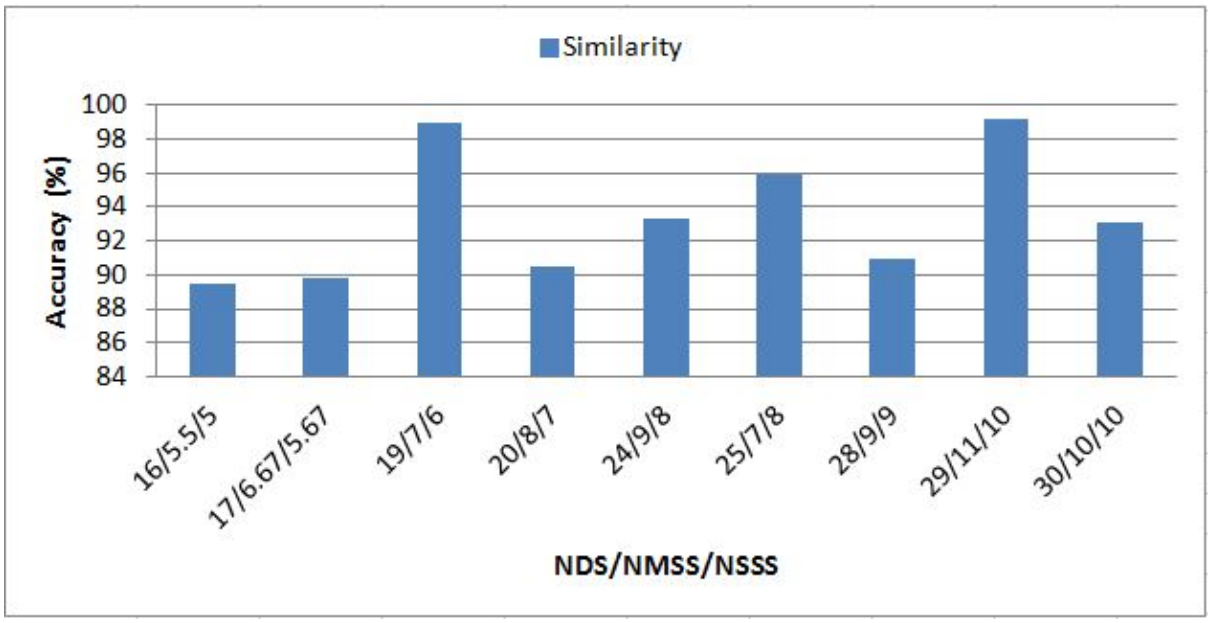

Figure 5.6: Similarity Graph of Experiment 2.

5.3.3 Experiment 3

This experiment is carried out in 12 different documents from datasets described in Section 5.1. Each document contains number of sentences more than 30. For the docunents having same number of sentence, the average value of each measure parameter is calculated and shown in table 5.4. This experiment shows that the average Precision, Recall and F-Score are obtained as 82.09% ,78.21% and 79.96% respectively. And, the average cosine similarity between manual and system summary is obtained as 96.05%. Figure 5.7 shows the graph between Precision, Recall and F-Score measures with respect to number of sentences in the documents,manual summary and system summary. And, Figure 5.8 shows the graph of manual and system summary similarity with respect to number of sentences in the documents,manual summary and system summary.

Table 5.4: Result of Experiment 3.

No. of Docs	Avg. NDS	Avg. NMSS	Avg. NSSS	Avg. MSP	Avg. SSP	Precission (%)	Recall (%)	F-score (%)	Similarity (%)
1	33	12	11	36.36	33.33	90.91	83.33	86.96	95.65
1	35	11	11	31.42	31.42	83.33	90.91	86.96	100
1	38	13	13	34.21	34.21	76.92	76.92	76.92	94.22
1	45	16	15	35.55	33.33	80	75	77.42	95.44
1	47	17	16	36.17	34.04	87.5	82.35	84.85	95.61
2	53	17.5	17	33.96	32.07	85.3	82.84	83.72	97.02
1	57	22	19	38.59	33.33	89.47	77.27	82.93	96.34
1	68	24	22	35.29	32.35	86.36	79.17	82.61	96.55
1	71	22	23	30.98	32.39	65.22	68.67	66.67	93.13
1	75	26	25	34.66	33.33	72	69.23	70.59	95.12
1	89	34	29	38.2	32.58	82.76	70	76.19	96.44

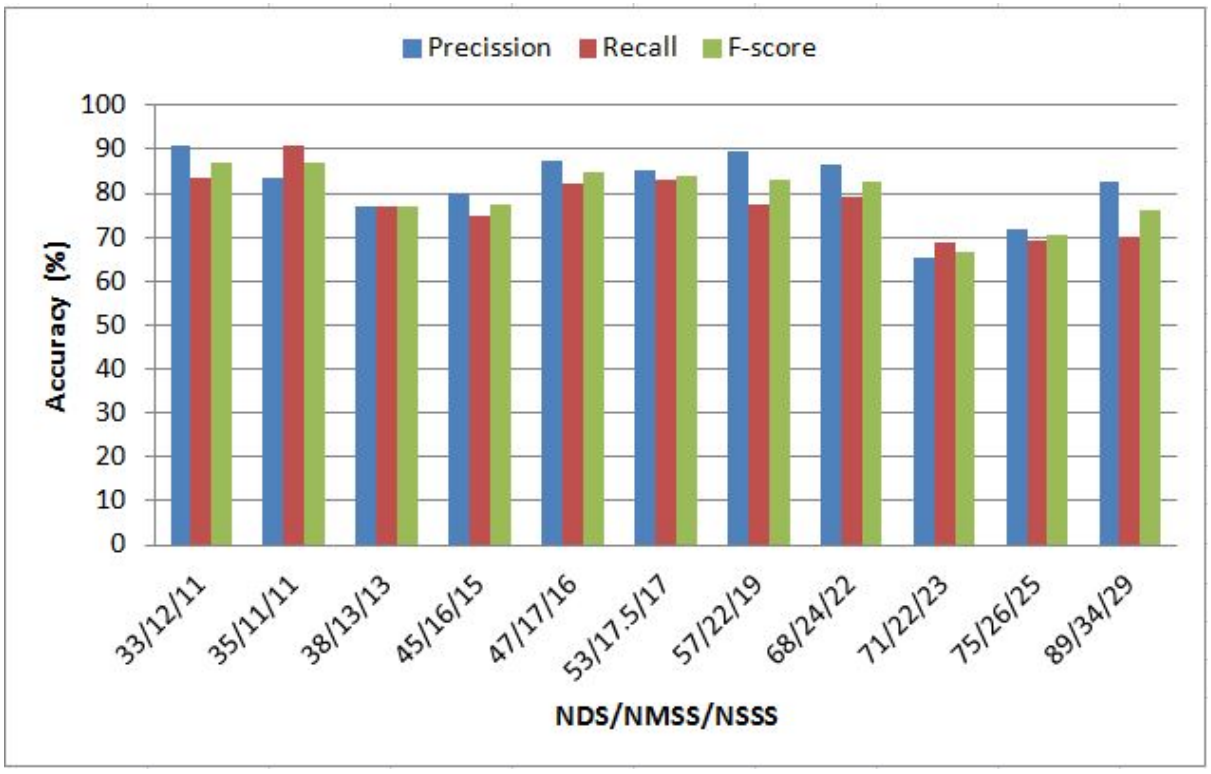

Figure 5.7: Precision-Recall-FScore Graph of Experiment 3.

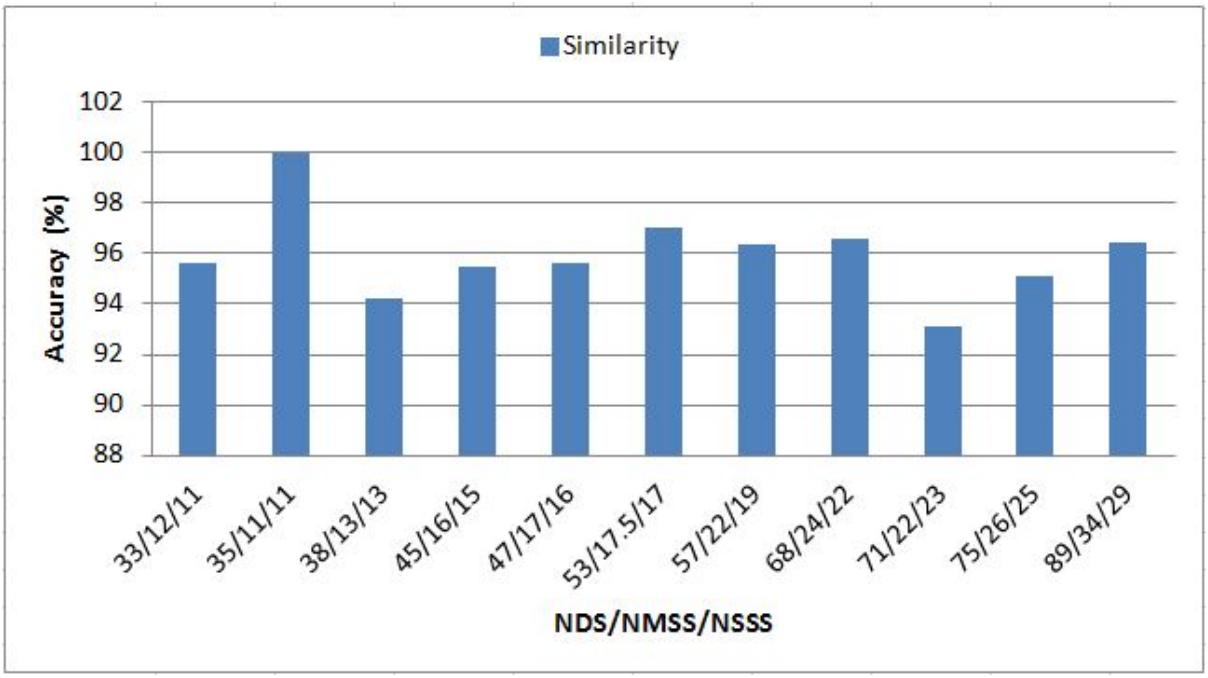

Figure 5.8: Similarity Graph of Experiment 3.

5.3.4 Overall System Performance and Result Analysis

The overall performance of the system is calculated by taking the average value of all the above three experiments. The overall Precision, Recall and F-Score of the system are 79.18%, 71.77% and 75.02% respectively. And, the average cosine similarity between manual and system summary is obtained as 91.16%. Figure 5.9 shows the graph between the overall Precision, Recall, F-Score and Similarity measures with respect to the number of sentences in the documents,manual summary and system summary.

Table 5.5: Average of System Measures.

Experiment No.	Avg. NDS	Avg. NMSS	Avg. NSSS	Avg. MSP	Avg. SSP	Precission (%)	Recall (%)	F-score (%)	Similarity (%)
Experiment # 1	10.86	4.21	3.57	40.06	33.24	75.72	64.05	68.99	84.89
Experiment # 2	21.50	7.67	7.08	36.19	33.02	79.74	73.06	76.11	92.54
Experiment # 3	55.33	19.33	18.17	34.79	32.87	82.09	78.21	79.96	96.05
Average	**29.23**	**10.40**	**9.61**	**37.01**	**33.04**	**79.18**	**71.77**	**75.02**	**91.16**

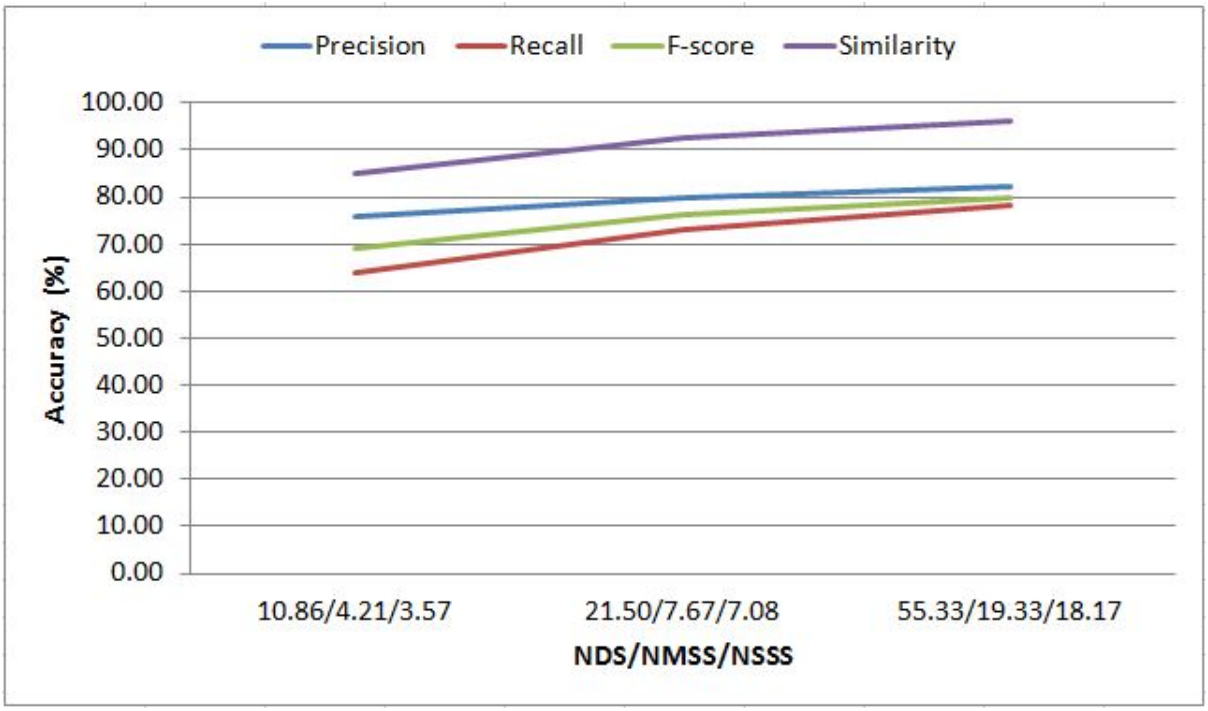

Figure 5.9: Graph of Overall Precision-Recall-FScore-Similarity.

From the above Figure 5.9, it is seen that the system performance improves as we increase the input document size. Similarity measure shows higher value than other three measures because some of the manual summaries are abstractive rather than extractive.

For the better accuracy of the system, better data dictionaries of stop words, symbols and NE are recommended along with some tuning in preprocessing steps. In addition to this, feature extraction can be made better by adding some domain specific knowledges and sentence extraction mechanism can be made more sophisticated by adding some learning mechanisms.

Chapter 6

CONCLUSION

6.1 Conclusion

In this dissertation work, an automatic text summarization for Nepali language based on sentence extraction for single document has been introduced. There are many practical importances of automatic text summarization in various fields. It can be used for note taking, information retrieval, data mining, indexing and many more.

The proposed method is mainly based on sentence scoring method. Each input sentences are passed through various stages of preprocessing and feature extraction to calculate sentence weights. Summery is generated by selecting most informative sentences from input document based on their weights.

The system deals with extracted information which increase the capability by giving weights to sentences that have words with the same meaning. On the other hand this process will increase the cohesion between sentences in the extracted summary. Besides this, an entity recognition module is applied on the documents to identify the relevance of the document.

For the evaluation of the proposed system, number of experiments are conducted on different datasets. Automated evaluation techniques are used in each experiments to validate the proposed system against the manual summaries. Different system evaluation matrices such precision, recall, f-score and cosine similarity are used to check the system accuracy. Empirical results shows that the overall accuracy of the system is achieved as 79.18% precision, 71.77% recall and 75.02% F-Score. Also, Cosine similarity measure gives overall similarity of 91.16% between manual summary and system summary.

Finally, our system is optimized, easy to use, general to any domain area and able to produce summaries comparable to human generated summaries. We expect the system to be used for a wide range of applications.

6.2 Limitations and Future Scope

In this work, the system can summarize a single Nepali text document which is solely based on sentence extraction model. So, sometimes there may be some semantic meaning loss in the summary version. If we include the semantic model in this work then the system will produce

more accurate summary as generated by human being, i.e abstract summary. This work can also be extended to summarize the multiple documents.

The performance of the proposed system may further be improved by improving stemming process along with other document pre-processing techniques. Exploring more number of features and applying learning algorithm for effective feature combination can also improve summary generation.

One problem with extracted sentences, they may contain anaphora links to the rest of the text. This has been investigated by [27]. Several heuristics have been proposed to solve this problem such as including the sentence just before the extracted one. Anaphora solving seems to be interesting point of research in future.

Traditionally, more than one reference summaries are used for evaluating each system generated summary, but in our work, we have used only one reference summary for summary evaluation. In future, one could consider more than one reference summaries for summary evaluation.

Our proposed system is tested with limited datasets as well as data dictionaries. For the better generalization of the system, larger datasets and sophisticated data dictionaries can be used for the evaluation of the system.

References

[1] E. Hovy, "Text summarization," in *The Oxford Handbook of Computational Linguistics* (R. Mitkov, ed.), Oxford Handbooks in Linguistics, ch. 32, pp. 583–598, Oxford: Oxford University Press, 2003.

[2] G. S. L. Vishal Gupta, "A survey of text summarization extractive techniques," vol. 2, no. 3, 2010.

[3] K. Owczarzak and H. T. Dang, "TAC 2010 Guided Summarization Task Guidelines," 2010.

[4] C. Kruengkrai and C. Jaruskulchai, "Generic text summarization using local and global properties of sentences," in *Web Intelligence*, pp. 201–206, IEEE Computer Society, 2003.

[5] B. K. Bal, "Towards building advanced natural language applications: An overview of the existing primary resources and applications in nepali," in *Proceedings of the 7th Workshop on Asian Language Resources*, ALR7, (Stroudsburg, PA, USA), pp. 165–170, Association for Computational Linguistics, 2009.

[6] H. P. Luhn, "Automatic creation of literature abstracts," *IBM Journal of Research and Development*, vol. 2, pp. 159–165, 1958.

[7] P. B. Baxendale, "Man-Made Index for Technical Literature - an Experiment," *IBM Journal of Research and Development*, vol. 2, no. 4, pp. 354–361, 1958.

[8] C.-Y. Lin and E. H. Hovy, "Identifying topics by position," in *ANLP*, pp. 283–290, 1997.

[9] Edmundson, "New methods in automatic extracting," *JACM: Journal of the ACM*, vol. 16, 1969.

[10] R. Al-Hashemi, "Text summarization extraction system (TSES) using extracted keywords," *Int. Arab J. e-Technol*, vol. 1, no. 4, pp. 164–168, 2010.

[11] K. Sarkar, "Bengali text summarization by sentence extraction," *CoRR*, vol. abs/1201.2240, 2012.

[12] N. M. Hewahi and K. A. Kwaik, "Automatic arabic text summarization system (AATSS) based on semantic features extraction," *IJTD*, vol. 3, no. 2, pp. 12–27, 2012.

[13] Z. B. Md. Majharul Haque, Suraiya Pervin, "Literature review of automatic single document text summarization using nlp," vol. 3, pp. 857–865, July 2013.

[14] Z. B. Md. Majharul Haque, Suraiya Pervin, "Literature review of automatic multiple documents text summarization," vol. 3, pp. 121–129, May 2013.

[15] H. Jing, "Sentence reduction for automatic text summarization," in *Proceedings of the Sixth Conference on Applied Natural Language Processing*, ANLC '00, (Stroudsburg, PA, USA), pp. 310–315, Association for Computational Linguistics, 2000.

[16] A. Patel, T. Siddiqui, and U. S. Tiwary, "A language independent approach to multilingual text summarization," in *Large Scale Semantic Access to Content (Text, Image, Video, and Sound)*, RIAO '07, (Paris, France, France), pp. 123–132, LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2007.

[17] McKeown, Kathleen, Radev, and D. R., "Generating summaries of multiple news articles," in *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Text Summarization, pp. 74–82, 1995.

[18] D. R. Radev and K. R. McKeown, "Generating natural language summaries from multiple on-line sources," *Comput. Linguist.*, vol. 24, pp. 470–500, Sept. 1998.

[19] J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz, "Multi-document summarization by sentence extraction," in *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic Summarization - Volume 4*, NAACL-ANLP-AutoSum '00, (Stroudsburg, PA, USA), pp. 40–48, Association for Computational Linguistics, 2000.

[20] J. Fukumoto and T. Sugimura, "Multi-document summarization using document set type classification."

[21] D. K. Evans, "Similarity-based multilingual multi-document summarization."

[22] B. K. Bal and P. Shrestha, "A morphological analyzer and a stemmer for nepali," *PAN Localization, Working Papers*, 2006.

[23] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, no. 5, pp. 513–523, 1988.

[24] J. L. Neto, A. Santos, C. Kaestner, and A. Freitas, "Document clustering and text summarization," in *Proc. 4th International Conference Practical Applications of Knowledge Discovery and Data Mining (PADD-2000)* (N. Mackin, ed.), (London), pp. 41–55, The Practical Application Company, 2000.

[25] J. Steinberger and K. Ježek, "Using latent semantic analysis in text summarization and summary evaluation," in *In Proc. ISIM '04*, pp. 93–100, 2004.

[26] S. Bam, "Named Entity Recognition for Nepali Text using Support Vector Machine," Master's thesis, Central Dept. of Computer Science and information Technology, T.U., Kitipur, Nepal, 2012.

[27] C. D. Paice, "Constructing literature abstracts by computer: Techniques and prospects," *Inf. Process. Manage*, vol. 26, no. 1, pp. 171–186, 1990.

Appendix A

Sample Source Codes

Symbols removal module

```
%public List<String> symbolRemoval(List<String> symbolDict,List<String>
    inputArray) {
        List<String> output = remove(symbolDict, inputArray);
        return output;
}
```

Stopword removal module

```
%public List<String> stopwordRomoval(List<String> stopwordDict,
        List<String> input) {
        List<String> output = remove(stopwordDict, input);
        return output;
}
```

remove Module

```
%public List<String> symbolRemoval(List<String> symbolDict,List<String>
    inputArray) {
        List<String> output = remove(symbolDict, inputArray);
        return output;
}
```

Stemming Module

```
%public class StemmerStart {
        static String tag_input, r1, p;
        static int bk, cm, rep = 0, i = 0, k = 0, repc = 0;
        public static List<String> stemmer(String input) {

        List<String> s_output = new ArrayList<String>();
        int index = 0;
        Date d = new Date();
```

```
d.getTime();
String unrecog = "\n";

Morph m = new Morph();
try {
tag_input = input;
} catch (Exception e) {
System.out.println(e.toString());
}
Tokenizer w = new Tokenizer();
w.tokenize(tag_input);

String out = w.getTokenizedOutput();
ReadWriteXML r = new ReadWriteXML(out);
cm = r.getNodeListLength();

for (int c = 0; c < cm; c++) {
String n = r.getTextCont(c);
if (!n.equals("nocontent")) {
m.setPMorph_number(0);
m.setSMorph_number(0);
m.setRootPos(null);
rep = 0;
m.setSecParse(rep);
p = "yes";
r1 = n;
i = 0;
k = 0;
while (!r1.isEmpty()) {
if (m.isRoot(r1) && p.equals("yes")
|| (m.isAltRoot(r1) && i > 0)) {

if (m.getSuffixExist(r1).equals("N") && i > 0) {
p = "no";
} else {
if (m.isAltRoot(r1) && i > 0)
m.setRoot(m.getAltRoot(r1));
else
m.setRoot(r1);
break;
}
}
else if (m.isRoot(r1 + "\u094d")) {
i++;
m.setSMorph_number(i);
m.isASuffix(r1);
r1 = m.getRoot();
```

```
}
else if (r1.endsWith("\u094d") && m.isRoot(r1.substring(0,
    r1.length() - 1))) {
r1 = r1.substring(0, r1.length() - 1);
m.setRoot(r1);
} else if (m.suffixPresent(r1, i)) {
i++;
m.setSMorph_number(i);
m.stripSuffix(r1);

r1 = m.getRoot();

} else if (m.prefixPresent(r1)) {
k++;

m.setPMorph_number(k);
m.stripPrefix(r1);
r1 = m.getRoot();
} else

if (k > 0 && i > 0) {
String tm, tmp;
String[] a = m.getPMorph();
for (int k1 = k; k1 > 0; k1--) {
tmp = r1;
for (int l = i; l > 0; l--) {
tm = m.generateWord(tmp, l);
if (m.isRoot(tm) || (m.isAltRoot(tm) && i > 0)) {
bk = 1;
m.setSMorph_number(l - 1);
m.setPMorph_number(k);
if ((m.isAltRoot(tm) && i > ))
m.setRoot(m.getAltRoot(tm));
else
m.setRoot(tm);
r1 = m.getRoot();

break;
} else {
bk = 0;
tmp = tm;
}

}
if (bk == 1) {
break;
}
```

```
if (k1 > 1)
r1 = a[k1] + r1;

}
if (bk != 1) {
r1 = "unrecognized";
k = 0;
}

} else {
m.setRoot("unrecognized");

}

if (m.getRoot().equals("unrecognized")) {
rep++;

int repeat = 0;
if (rep == 1) {

for (int l = i; l > 0; l--) {
if (m.isRepeat(Integer.toString(m.getSMorph_rulenum(l)))) {
repeat = 1;
break;// for any suffix that has a repeat
} else
repeat = 0;

}
}
if (repeat == 1) {
r1 = n;
m.setPMorph_number(0);
m.setSMorph_number(0);
i = 0;
k = 0;
p = "yes";

m.setSecParse(rep);
} else
break;

}

}
if (m.getRoot().equals("unrecognized")) {
m.handleDublicateWord(n);
```

```
}
if (m.getRoot().equals("unrecognized")) {

m.handleCompoundWord(n);
}

if (m.getRoot().equals("unrecognized")) {
unrecog = unrecog + n + "\n";
}

StringBuffer a = new StringBuffer("");
StringBuffer mspos = new StringBuffer("");
StringBuffer mppos = new StringBuffer("");
StringBuffer b = new StringBuffer("");
String[] pmorpheme = m.getPMorph();
String[] pdt = m.getPDes();
String[] smorpheme = m.getSMorph();
String[] sdt = m.getSDes();

int p = m.getPMorph_number();
int s = m.getSMorph_number();

for (int j = 0; j < p; j++) {
b.append(pmorpheme[j + 1] + "(" + pdt[j + 1] + ")" + "+");
mppos.append(pdt[j + 1] + "_");
}
for (int j = s; j > 0; j--) {
a.append("+" + smorpheme[j] + "(" + sdt[j] + ")");
mspos.append("_" + sdt[j]);
}
r.setMorphContent(c, b + m.getRoot() + "(" + m.getRootPos()
+ ")" + a);

if (!m.getRoot().equals("unrecognized")) {
r.setPosContent(c, mppos + m.getRootPos() + mspos);
}

if (m.getRoot().matches("unrecognized"))
s_output.add(n);
else
s_output.add(m.getRoot());
index = index + 1;
}
}
Date d1 = new Date();
d1.getTime();
```

```
        return s_output;

        }
        }
```

Position feature module

```
%public double sentencePosition(List<String> input,int sPos, String
    sentence) {
        double noOfSentences = input.size();
        if (noOfSentences == 0) {
                return 0.0;
        }
        double absSPos = ((noOfSentences - sPos + 1) / noOfSentences);
        return absSPos;
}
```

Length feature module

```
%public double lengthOfSentence(int noOfWordsInLongestSentence,
        String sentence) {
        int noOfWordsInSentence = noOfWords(sentence);
        double output = (double) noOfWordsInSentence
                        / noOfWordsInLongestSentence;
        return output;
}
```

Named Entity Recognition Module

```
%public int neClass(List<List<String>> neDict, String word) {
        if (neDict.get(0).contains(word)) {
                return 1;
        } else if (neDict.get(1).contains(word)) {
                return 2;
        } else if (neDict.get(2).contains(word)) {
                return 3;
        } else if (neDict.get(3).contains(word)) {
                return 4;
        } else {
                        return 5;
        }
}
```

Term frequency (TF X ISF) module

```
%
public int tf(List<String> tokens, String word) {
        int tf = Collections.frequency(tokens, word);
        return tf;
}

public int isf(List<String> sentences, String word) {
        int n = 0;
        for (Object o : sentences) {
                if (o.toString().contains(word)) {
                        n = n + 1;
                }
        }
        return n;
}

public double wordWeight(List<String> input, String word) {
        List<String> sentences = listSentencise(input); // sentence list
        List<String> tokens = listTokanize(input); // token list
        int ns = sentences.size(); // number of sentences
        int tf = tf(tokens, word);
        int isf = isf(sentences, word);
        double tf_isf = (double) tf * Math.log((double) ns / (double) isf);
        return tf_isf;
}
```

Sentence analysis module

```
%
public List<Double> sentenceWeight(List<String> input) {
        List<Double> weights = new ArrayList<Double>();
        List<Double> tw = new ArrayList<Double>();
        List<String> sentences = listSentencise(input); // sentence list
        List<String> tokens = listTokanize(input); // token list
        int ns = sentences.size(); // number of sentences
        for (Object o : sentences) {
                double wi = sentenceWeight2(sentences, tokens,
                   o.toString());
                tw.add(wi);
        }
        double maxw = Collections.max(tw);
        for (Object o : tw) {
                weights.add((Double) o / maxw);
        }
```

```
        return weights;
}

        public double sentenceWeight1(List<String> sentenceArray, String
            sentence) {
        List<String> tokenArray = listTokanize(sentenceArray);
        double wi = 0.0;
        List<String> sentenceTokenArray = sentenceTokanize(sentence);
        int nTokens = sentenceTokenArray.size();
        for (int i = 0; i < nTokens; i++) {
                double twi = 0.0;
                String token = sentenceTokenArray.get(i).toString();
                twi = wordWeight1(sentenceArray, tokenArray, token);
                wi = wi + twi;
        }
        return wi;
}

public double sentenceWeight2(List<String> sentences, List<String> tokens,
        String sentence) {
        double wi = 0.0;
        String[] tt = sentence.toString().split("[ ,\t]");
        for (int i = 0; i < tt.length; i++) {
                double twi = 0.0;
                if (!tt[i].trim().isEmpty()) {
                        twi = wordWeight1(sentences, tokens, tt[i]);
                        wi = wi + twi;
                }
        }
        return wi;
}

public List<String> listTokanize(List<String> input) {
List<String> temp = new ArrayList<String>();
        for (Object o : input) {
                String[] tt = o.toString().split("[ ,?|\t\n]");
                for (int i = 0; i < tt.length; i++) {
                        if (!tt[i].trim().isEmpty()) {
                                temp.add(tt[i]);
                        }
                }
        }
        return temp;
}

public List<String> sentenceTokanize(String sentence) {
        List<String> temp = new ArrayList<String>();
```

```
        String[] tt = sentence.split("[ ,?|\t\n]");
        for (int i = 0; i < tt.length; i++) {
                if (!tt[i].trim().isEmpty()) {
                        temp.add(tt[i]);
                }
        }
        return temp;
}

public List<String> listSentencise(List<String> input) {
        List<String> temp = new ArrayList<String>();
                for (String s : input) {
                if(s==null||s.length()==1||s.isEmpty()||s.trim().equals("")
                ||s.equals("")){continue;
                }
                        String[] tt = s.trim().split("(?<=[?|]+)");
                        for (int i = 0; i < tt.length; i++) {
if (tt[i].length()!=0||!tt[i].trim().equals("")||!tt[i].trim().isEmpty()) {
                                        temp.add(tt[i].trim());
                                }
                        }
        }
        return temp;
}
```

Sumamry generation module

```
%public List<String> summaryGeneration(List<String> input, List<String>
    ppOutput,
        List<List<String>> neDict, float nSummary) {
        List<String> summary=new ArrayList<String>();
        double[][] scoreArray = sentenceScoring(ppOutput, neDict);
        double[][] rankArray = sentenceRanking(scoreArray);
        int nRows = rankArray.length;
        int nSummarySentences = (int)Math.round(nRows * nSummary);
        int[]ssIndexArry=new int[nSummarySentences];
        for (int i = 0; i < nSummarySentences; i++) {
                ssIndexArry[i]=(int) (rankArray[i][0]);
        }
        Arrays.sort(ssIndexArry);
        for (int i = 0; i < nSummarySentences; i++) {
                summary.add(input.get(ssIndexArry[i]));
        }
        return summary;
}
```

```
public double[][] sentenceRanking(double[][] scoreArray) {
        Arrays.sort(scoreArray, new Comparator<double[]>() {
        @Override
        public int compare(double[] int1, double[] int2) {
                Double numOfKeys1 = int1[1];
                Double numOfKeys2 = int2[1];
                return -numOfKeys1.compareTo(numOfKeys2); // to return
                    result in
        }
        });
        return scoreArray;
}

public double[][] sentenceScoring(List<String> input,
        List<List<String>> neDict) {
        double[][] featureArray = sentenceFeatures(input, neDict);
        int nSentences = input.size();
        double[][] scoreArray = new double[nSentences][2];
        for (int i = 0; i < nSentences; i++) {
        scoreArray[i][0] = i;
        scoreArray[i][1] = featureArray[i][1]
                        + featureArray[i][2] + featureArray[i][3]
                        + featureArray[i][4] + featureArray[i][5]
                        + featureArray[i][6] + featureArray[i][7]
                        + featureArray[i][8];
        }
        System.out.println("-----------------------------------------");
        System.out.println("Features of Sentences");
        System.out.println("-----------------------------------------");
        for (int i=0;i<nSentences;i++) {
        for (int j=0;j<9;j++){
                System.out.print(featureArray[i][j] +" ");
        }
        System.out.println();
        }
        System.out.println("-----------------------------------------");
        System.out.println("Score of Sentences");
        System.out.println("-----------------------------------------");
        for (int i=0;i<nSentences;i++) {
        for (int j=0;j<2;j++){
                System.out.print(scoreArray[i][j] +" ");
        }
        System.out.println();
        }
        return scoreArray;
}
```

```
public double[][] sentenceFeatures(List<String> input,  List<List<String>>
    neDict) {
        int nWordsInLongestSentence =
            feDao.noOfWordsInLongestSentence(input);
        int nSentences = input.size();
        double[][] featureArray = new double[nSentences][nFeatures];
        String sentence;
        List<String> tokenArray = new ArrayList<String>();
        for (int i = 0; i < nSentences; i++) {
        if (input.get(i).isEmpty() || input.get(i).trim().isEmpty()
                        || input.get(i) == null) {
                featureArray[i][0] = i;
                featureArray[i][1] = 0;
                featureArray[i][2] = 0;
                featureArray[i][3] = 0;
                featureArray[i][4] = 0;
                featureArray[i][5] = 0;
                featureArray[i][6] = 0;
                featureArray[i][7] = 0;
                featureArray[i][8] = 0;
        } else {
                sentence = input.get(i).toString().trim();
                featureArray[i][0] = i;
                featureArray[i][1] = feDao.lengthOfSentence(
                                nWordsInLongestSentence, sentence);

                featureArray[i][2] = feDao.sentencePosition(input,i+1,
                    sentence);

                tokenArray = feDao.sentenceTokanize(sentence);
                int nTokens = tokenArray.size();
                int neResult;
                String word;
                int nPersons = 0, nLocations = 0, nOrganizations = 0,
                    nMisc = 0, nNotNE = 0;
                for (int j = 0; j < nTokens; j++) {
                        word = tokenArray.get(j);
                        neResult = feDao.neClass(neDict, word);
                        switch (neResult) {
                        case 1:
                                nPersons = nPersons + 1;
                                break;
                        case 2:
                                nLocations = nLocations + 1;
                                break;
                        case 3:
                                nOrganizations = nOrganizations + 1;
```

```
                                break;
                        case 4:
                                nMisc = nMisc + 1;
                                break;
                        case 5:
                                nNotNE = nNotNE + 1;
                                break;
                        default:
                                break;
                        }

                        featureArray[i][3] = nPersons;
                        featureArray[i][4] = nLocations;
                        featureArray[i][5] = nOrganizations;
                        featureArray[i][6] = nMisc;
                        featureArray[i][7] = 0.0; //featureArray[i][7] =
                            nNotNE;
                }
                featureArray[i][8] = feDao.sentenceWeight1(input,
                    sentence);
        }
}
return featureArray;
}
```

System evaluation module

```
%
public void systemEvaluation(List<String> manualSummary,List<String>
    systemSummary) {
        double precision=precision(manualSummary, systemSummary);
        double recall=recall(manualSummary, systemSummary);
        double beta=1;
        double fscore=fscore(precision, recall, beta);

        System.out.println("-----------------------------------------");
        System.out.println("Results");
        System.out.println("\tPrecision:
            "+roundTwoDecimals(precision*100)+"%");
        System.out.println("\tRecall: "+roundTwoDecimals(recall*100)+"%");
        System.out.println("\tF-Score: "+roundTwoDecimals(fscore*100)+"%");
}

public String[] systemEvaluation1(List<String> manualSummary,List<String>
    systemSummary) {
        double precision=precision(manualSummary, systemSummary);
```

```
        double recall=recall(manualSummary, systemSummary);
        double beta=1;
        double fscore=fscore(precision, recall, beta);
        String[] result=new String[3];
        result[0]=roundTwoDecimals(precision*100)+"%".toString();
        result[1]=roundTwoDecimals(recall*100)+"%".toString();
        result[2]=roundTwoDecimals(fscore*100)+"%".toString();
        return result;
}

double roundTwoDecimals(double d) {
        DecimalFormat twoDForm = new DecimalFormat("#.##");
        return Double.valueOf(twoDForm.format(d));
}

public double precision(List<String> manualSummary,List<String>
    systemSummary) {
        int nSystemSumarySentences=systemSummary.size();
        Set<String> intersect = new HashSet<String>(manualSummary);
        intersect.retainAll(systemSummary);
        int nSystemCorrectSummarySentences=intersect.size();
        return
           (double)nSystemCorrectSummarySentences/nSystemSumarySentences;
}

public double recall(List<String> manualSummary,List<String>
    systemSummary) {
        int nManualSumarySentences=manualSummary.size();
        Set<String> intersect = new HashSet<String>(manualSummary);
        intersect.retainAll(systemSummary);
        int nSystemCorrectSummarySentences=intersect.size();
        return
           (double)nSystemCorrectSummarySentences/nManualSumarySentences;
}

public double fscore(double precision, double recall, double beta) {
        double f=0.0;
        if(precision!=0.0 && recall!=0.0){
                f = ((beta * beta + 1) * precision * recall)
                        / ((beta * beta) * precision + recall);
        }
        return f;
}

public class values
{
        int val1;
```

```
        int val2;
        values(int v1, int v2)
        {
                this.val1=v1;
                this.val2=v2;
        }
        public void Update_VAl(int v1, int v2)
        {
                this.val1=v1;
                this.val2=v2;
        }
}//end of class values

public double cosineSimilarityScore(String Text1, String Text2){
        double sim_score=0.0000000;
        String [] word_seq_text1 = Text1.split(" ");
        String [] word_seq_text2 = Text2.split(" ");
        Hashtable<String, values> word_freq_vector = new Hashtable<String,
           EvaluationDaoImpl.values>();
        LinkedList<String> Distinct_words_text_1_2 = new
           LinkedList<String>();
        for(int i=0;i<word_seq_text1.length;i++)
        {
                String tmp_wd = word_seq_text1[i].trim();
        if(tmp_wd.length()>0)
        {
        if(word_freq_vector.containsKey(tmp_wd))
        {
        values vals1 = word_freq_vector.get(tmp_wd);
        int freq1 = vals1.val1+1;
        int freq2 = vals1.val2;
        vals1.Update_VAl(freq1, freq2);
        word_freq_vector.put(tmp_wd, vals1);
        }
        else
        {
        values vals1 = new values(1, 0);
        word_freq_vector.put(tmp_wd, vals1);
        Distinct_words_text_1_2.add(tmp_wd);
        }
        }
        }

        for(int i=0;i<word_seq_text2.length;i++)
        {
        String tmp_wd = word_seq_text2[i].trim();
        if(tmp_wd.length()>0)
```

```
        {
        if(word_freq_vector.containsKey(tmp_wd))
        {
        values vals1 = word_freq_vector.get(tmp_wd);
        int freq1 = vals1.val1;
        int freq2 = vals1.val2+1;
        vals1.Update_VAl(freq1, freq2);
        word_freq_vector.put(tmp_wd, vals1);
        }
        else
        {
        values vals1 = new values(0, 1);
        word_freq_vector.put(tmp_wd, vals1);
        Distinct_words_text_1_2.add(tmp_wd);
        }
        }
        }

        double VectAB = 0.0000000;
        double VectA_Sq = 0.0000000;
        double VectB_Sq = 0.0000000;

        for(int i=0;i<Distinct_words_text_1_2.size();i++)
        {
        values vals12 =
           word_freq_vector.get(Distinct_words_text_1_2.get(i));

        double freq1 = (double)vals12.val1;
        double freq2 = (double)vals12.val2;

        VectAB=VectAB+(freq1*freq2);

        VectA_Sq = VectA_Sq + freq1*freq1;
        VectB_Sq = VectB_Sq + freq2*freq2;
        }
        sim_score = ((VectAB)/(Math.sqrt(VectA_Sq)*Math.sqrt(VectB_Sq)));

        return(roundTwoDecimals(sim_score*100));
        }
}
```

Appendix B

Sample Input and Output