Investigations on Effective Techniques for Bengali Information Retrieval and Summarization of Search Results

Thesis submitted by
Soma Chatterjee

DOCTOR OF PHILOSOPHY (Engineering)

Department of Computer Science and Engineering,
Faculty Council of Engineering & Technology,
Jadavpur University
Kolkata, India
2024

JADAVPUR UNIVERSITY

KOLKATA-700032, INDIA

INDEX NO. 207/17/E

1. Title of the Thesis:
   Investigations on Effective Techniques for Bengali Information Retrieval and Summarization of Search Results
2. Name, Designation & Institution of the Supervisor/s:

   (a) Prof. Kamal Sarkar
   Professor
   Department of Computer Science and Engineering
   Jadavpur University, Kolkata-700032, India

List of Publications

Papers in Journals

1. Soma Chatterjee and Kamal Sarkar (2018). Combining IR Models for Bengali Information Retrieval. International Journal of Information Retrieval Research (IJIRR). Vol.8(3), pp.68-83.

2. Soma Chatterjee and Kamal Sarkar (2019). A Comparative Study of Three IR Models for Bengali Document Retrieval. International Journal of Computer Sciences & Engineering (IJCSE). Vol.7(1), E-ISSN: 2347-2693.

3. Soma Chatterjee and Kamal Sarkar (2023). Improved Cluster-based Smoothing for Language Models Applied to Bengali Information Retrieval. International Journal of the Indian Academy of Sciences, SADHANA, 48(4), pp.211.

Papers in Conference Proceedings

1. Soma Chatterjee and Kamal Sarkar (2022). Predicting Word Importance Using a Support Vector Regression Model for Multi Document Text Summarization. In: International conference on Advance in Data–driven Computing and Intelligent System (ADCIS-2022), pp.83-97.

2. Soma Chatterjee and Kamal Sarkar (2022). Bengali Document Retrieval Using Model Combination. In: 3rd International Conference on Frontiers in Computing and Systems(COMSYS-2022), pp. 91-101.

3. Soma Chatterjee and Kamal Sarkar (2023). A Hybrid Query Expansion Method for Effective Bengali Information Retrieval. In: 4th International Conference on Frontiers in Computing and Systems(COMSYS-2023). (Accepted and Presented).

List of Presentations in National/International Conference:

1. Soma Chatterjee and Kamal Sarkar(2018). A Comparative Study of Three IR Models for Bengali Document Retrieval. In 1st International conference on innovations in Computer Science (ICICS-2018) at The University of Burdwan, WB, during December 21-22, 2018.

2. Soma Chatterjee and Kamal Sarkar(2022). Predicting Word Importance Using a Support Vector Regression Model for Multi Document Text Summarization. In: International conference on Advance in Data–driven Computing and Intelligent System (ADCIS-2022).BITS Pilani, K K Birla Goa Campus, during September 23-25, 2022.

3. Soma Chatterjee and Kamal Sarkar(2022). Bengali Document Retrieval Using Model Combination. In: 3rd International Conference on Frontiers in Computing and Systems(COMSYS-2022).IIT Ropar, Punjab, during December 19-21,2022.

4. Soma Chatterjee and Kamal Sarkar (2023). A Hybrid Query Expansion Method for Effective Bengali Information Retrieval. In: 4th International Conference on Frontiers in Computing and Systems(COMSYS-2023). Indian Institute of Technology Mandi, Himachal Pradesh, during 16th - 17th October 2023. .

STATEMENT OF ORIGINALITY

I Soma Chatterjee registered on 12/09/2017 do hereby declare that this thesis entitled ”Investigations on Effective Techniques for Bengali Information Retrieval and Summarization of Search Results” contains a literature survey and original research work done by the undersigned candidate as part of Doctoral studies.

All information in this thesis has been obtained and presented in accordance with existing academic rules and ethical conduct. I declare that, as required by these rules and conduct, I have fully cited and referred all materials and results that are not original to this work.

I also declare that I have checked this thesis as per the “Policy on Anti Plagiarism, Jadavpur University, 2019”, and the level of similarity as checked by iThenticate software is 8%.

Soma chatterjee

Signature of the Candidate:

Date: 05.01.2024

05.01.2024

Certified by Supervisor:
(Signature with date,seal)

Professor
Computer Sc. & Engg. Department
Jadavpur University
Kolkata-700032

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the thesis entitled **Investigations on Effective Techniques for Bengali Information Retrieval and Summarization of Search Results** submitted by Soma Chatterjee, who got his name registered on 12.09.2017 for the award of Ph.D. (Engineering) degree of Jadavpur University is absolutely based upon his own work under the supervision of Prof. Kamal Sarkar, Department of Computer Science and Engineering, Jadavpur University, Kolkata-700032, India, and neither his thesis nor any part of the thesis has been submitted for any degree/diploma or any other academic award anywhere before.

KSarkar 05/01/2024

....

Signature of the Supervisor
and date with Office Seal

Professor
Computer Sc. & Engg. Department
Jadavpur University
Kolkata-700032

Dedication

To my family and my mother

Acknowledgements

Finally, it is time for me to acknowledge all those who inspired me, supported me and helped me to get to the place where I am today.

I take this opportunity to express a deep sense of gratitude to Prof. (Dr.) Kamal Sarkar for their supervision and invaluable co-operation. This thesis would not have been possible without their constant inspiration and unbelievable support of them over the last few years.

I have had an amazing group of Labmates. Each of them deserves my gratitude: Sohini Roy Chowdhury, Kakuli Mishra Ashmita Dey, and Dr.Sagnik Sen. Thank you to all of you for the unselfish help, insights, and feedback, and for making the science a collaborative effort. With that, a special thanks to Prof. Kamal Sarkar. The Journey would remain incomplete without their suggestions, unconditional help, and encouragement.

I would also like to thank all of my colleagues from the Computer Science and Engineering Department, at Jadavpur University for providing me with a friendly research environment.

Finally, most important of all, I would like to dedicate the thesis to my mother Late Mrs. Chameli Sarkar Dey, and my brother Mr. Supriya Sarkar, to honor their love, patience, encouragement, and support during my research. I would also express my appreciation to my daughter Miss. Samarpita Chatterjee and my husband Mr. Sumit Chatterjee for their unwavering support and love.

Soma Chatterjee 05.01.2024

Date: 05.01.2024

(Soma Chatterjee)

Abstract

In Today's world, a large percentage of internet users rely on search engines to find relevant information rapidly. Most internet search engines use classical information retrieval techniques for retrieving documents in response to a natural language query. People commonly use popular search engines such as Google, Yahoo, DuckDuckGo, Baidu, etc. Before the advent of WWW(World Wide Web), retrieval systems were used to manage information in libraries, government organizations, and archiving. In these cases, the professional indexers acted as an intermediary for the common users and they could identify the customer needs through an interactive dialogue with the customers because they knew the collection, and document representation, and they were experts in formulating Boolean search operators to retrieve the documents. However, information retrieval systems used by the various modern search engines are designed for naive users unfamiliar with the collection, document representation, and formulation of queries. Most existing IR systems were designed for retrieving English documents in response to an English query.

Nowadays non-English content is rapidly increasing on the Internet. Therefore, the major search engines are trying to add support for the Indian languages including Bengali. Although most search engines use basic IR techniques, the application of IR techniques designed to focus on a certain target language can help in improving retrieval performance. Compared to English, the Bengali language has a more complex morphology. Bengali is a language in which verbs and nouns have many inflected forms. In the Bengali language, proper nouns can appear in various forms, for example, Ratan-i, Ratan-o, and Ratan-er. Proper nouns can also have more complex morphology where multiple atomic suffixes are added to proper nouns, Ratan-der-ke-o. The singular and plural forms of nouns can have inflection like aam-guli, jamider-er. Compared to nouns, verbs are more inflected in Bengali. Compound words are very frequently used in the Bengali language. Compound words containing verbs, adjectives, and nouns can have many morphological variations. Besides this morphological variation, like in English, new words can be formed by derived morphology. Due to the rich morphology of the Bengali language, the term mismatching problem becomes more critical while using Bengali queries for searching.

Although research on Bengali information retrieval had started in the long past, the research progress had been a little bit slow. Most earlier studies on Bengali IR focused on morphological analysis of the terms and finding roots or stems of the words using a lemmatizer or stemmer for effective indexing of the Bengali texts and integrating it with the traditional IR models such as the TF-IDF model, Okapi BM25 model. In recent times, an effective Deep Learning technique known as transfer learning has transformed and revolutionized the field of natural language processing. Several transfer learning techniques and architectures have been developed to advance the state-of-the-art on a variety of NLP tasks. The pre-trained word embeddings have been more popular due to the widespread availability and simplicity of integration with a new task. The use of pre-trained word embeddings in word and document representation can be effective for dealing with the word mismatch problem which is a critical problem for Bengali IR caused by morphological variants. To date, much research work has not been done for Bengali information retrieval to date and research on Indian languages is not at par with the English language also. Motivated by this, in this thesis, we investigate novel algorithms for Bengali IR. We have proposed several techniques for improving the Bengali IR model in various ways (1) Enhancing the smoothing process for a document language model, (2) semantic matching-based retrieval using large language model(LLM)-based word or document representation, (3) query expansion using pre-trained word embedding, pseudo-relevance feedback, or user feedback or hybrid methods, and (4) combining multiple IR models. We have evaluated the proposed approaches using benchmark IR datasets for Bengali. The experimental results and analyses establish that our proposed methods are effective for Bengali information retrieval.

Since an IR model returns a large number of documents, to aid the users in spotting the relevant information quickly, we propose a method for clustering the search results and producing summaries of the clusters so that the user can quickly identify the cluster containing the most relevant document by reading the summaries. We have proposed a new method for multi-document summarization which is used for creating a summary for each cluster. For clustering search results, we have used a histogram-based clustering method enhanced using word embedding. The multi-document summarization method has also been tested on a benchmark summarization dataset for English and a Bengali dataset developed by us. The experimental results and analyses establish that the proposed multi-document summarization method is effective for creating cluster summaries. It is also established that clustering and summarization of search results are useful for quickening to find the relevant information.

Keywords: Bengali Language, Information Retrieval, Smoothing, Document Language Model, Semantic Matching-based Retrieval, Word Embedding, Query Expansion, Pseudo-relevance Feedback, Hybrid Query Expansion, Model Combination, Clustering, Multi-document Summarization

Contents

List of Figures

List of Tables

1
Introduction and Scope of the Thesis

1.1 Introduction

In Today's world, we can not imagine a life without the Internet. To cope with our fast-paced world, we need the internet. Surveys show that a large percentage of internet users rely on search engines to find relevant information rapidly. Most internet search engines use classical information retrieval techniques for retrieving documents in response to a natural language query. People commonly use popular search engines such as Google, Yahoo, DuckDuckGo, Baidu, etc.

Before the advent of WWW(World Wide Web), retrieval systems were used to manage information in libraries, government organizations, and archiving. In these cases, the professional indexers acted as an intermediary for the common users and they could identify the customer needs through an interactive dialogue with the customers because they knew the collection, and document representation, and they were experts in formulating Boolean search operators to retrieve the documents.

However, information retrieval systems used by the various modern search engines are designed for naive users unfamiliar with the collection, document representation, and formulation of queries. The key requirements for these kinds of IR systems are (1)the ability to process natural language queries, (2)ranking the retrieved documents by their estimated relevance scores, and (3) automatic query reformulation for refining the search results. Most existing IR systems were designed for retrieving English documents in response to an English query.

Although IR systems help to reduce information overload problems by quickly finding relevant documents, they raise a new problem by returning too many documents for a single query. To pinpoint which documents are more useful, the user often has to scan through a large number of documents to find out a few of them that satisfy their information needs. This is also a tedious task. The early research results[3] have shown that a search engine could be more effective if the returned documents are grouped into clusters and summary information could be provided to help the users explore the retrieved results efficiently.

Clustering search results and summarising each cluster makes information more easily understandable to users and makes it easier for them to find the information they need quickly. It is required to summarise a group of documents or a single one to deal with the problem of information overload. If there is a summary for the entire cluster, the user can determine the cluster relevance by simply looking at the summary. The user can choose which cluster might include the pertinent documents of his or her interest by quickly perusing the cluster-wise summaries. Users may decide to open the clusters and review the summaries of the multiple documents inside the cluster once the pertinent cluster has been chosen to identify the pertinent document within the cluster. Thus total searching time can be reduced. Therefore, there is a need for an IR system that can do effective retrieval of relevant documents and produce summaries of the retrieved documents.

1.2 Basic Definitions

1.2.1 Information Retrieval

Information retrieval is defined by Manning et al. as ”Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)”[4].

1.2.2 Text Summarization

Automatic text summarization is an area that deals with finding the appropriate algorithms for generating a condensed version of a document or a document set. Radev et. al. has defined text summarization as ”A summary can be loosely defined as a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually significantly less than that. Text here is used rather loosely and can refer to speech, multimedia documents, hypertext, etc”[5].

1.3 Bengali Monolingual IR and Summarization of search results

Although the English language has become the universal language of the Internet as well as the traditional search engines, nowadays non-English online users are rapidly increasing. The article published in Economic Times on April 30, 2017, had the caption “536 million users to log onto the Internet in Indian languages by 2021”. According to this article, Google-KPMG estimated that 536 million Indian users would use regional languages on the Internet by 2021. This article also reports that along with Hindi users, Marathi, Bengali, Tamil, and Telugu users are expected to drive the volume growth of online Indian language users.

Therefore, the major search engines are trying to add support for the Indian languages including Bengali. Although most search engines use basic IR techniques, the application of IR techniques designed to focus on a certain target language can help in improving retrieval performance. Compared to English, the Bengali language has a more complex morphology. Bengali is a language in which verbs and nouns have many inflected forms. In the Bengali language, proper nouns can appear in various forms, for example, Ratan-i, Ratan-o, and Ratan-er. Proper nouns can also have more complex morphology where multiple atomic suffixes are added to proper nouns, Ratan-der-ke-o. The singular and plural forms of nouns can have inflection like aam-guli, jamider-er. Compared to nouns, verbs are more inflected in Bengali. Compound words are very frequently used in the Bengali language. Compound words containing verbs, adjectives, and nouns can have many morphological variations. Besides this morphological variation, like in English, new words can be formed by derived morphology. Although research on Bengali information retrieval had started in the long past, the research progress had been a little bit slow before FIRE(Forum of Information Retrieval) conducted a shared task on Ad hoc Bengali monolingual retrieval and released datasets in 2008. Thereafter, a number of shared tasks on Ad hoc Bengali monolingual retrieval were conducted in successive years and datasets were released by FIRE. This led the researchers to focus on Bengali monolingual retrieval.

Most earlier studies using FIRE Bengali IR datasets focused on morphological analysis of the terms and finding roots or stems of the words using lemmatizer or stemmer for effective indexing of the Bengali texts[6, 7]. Although some IR models such as the TF-IDF model, Okapi BM25 model, word embedding-based IR MODEL, hybrid model, [1, 8, 9], were developed and tested on FIRE datasets for proving their effectiveness. The research works on clustering and summarizing Bengali search results are hardly found in the literature. Moreover, deep learning-based word and document representation using the Word2Vec model or the transformer-based model are not much studied for Bengali monolingual retrieval.

As mentioned earlier, due to the rich morphology of the Bengali language, the term mismatching problem becomes more critical while using Bengali queries for searching. The deep learning-based model for word and document representation can be effective for dealing with the word mismatch problem. Since much research work has not been done for Bengali information retrieval to date, there are scopes for improvements in the three basic components of IR systems: query processing, ranking, and query expansion using WordNet, pseudo-relevance feedback, or user feedback or hybrid methods. In this research work, we have contributed to these components for developing a Bengali information retrieval. To aid the users in spotting the relevant information, the search results are clustered and summaries of the clusters are generated to present to the users.

Although the research on Bengali information retrieval and summarizing search results is not limited, we have surveyed the major approaches to Bengali IR and summarization. First, the major approaches to Bengali information retrieval are discussed, and then approaches to the summarization of search results are discussed.

1.4 Major approaches to Bengali Information Retrieval

Over the last several decades, many researchers have proposed a number of retrieval approaches and we have classified major approaches as follows:

- **Vector Space Approach**
- **Probabilistic Approach**
- **Semantic Approach**
- **Query Expansion**

Figure 1.3 shows the major approaches to Bengali information retrieval approaches and important IR models under each approach.

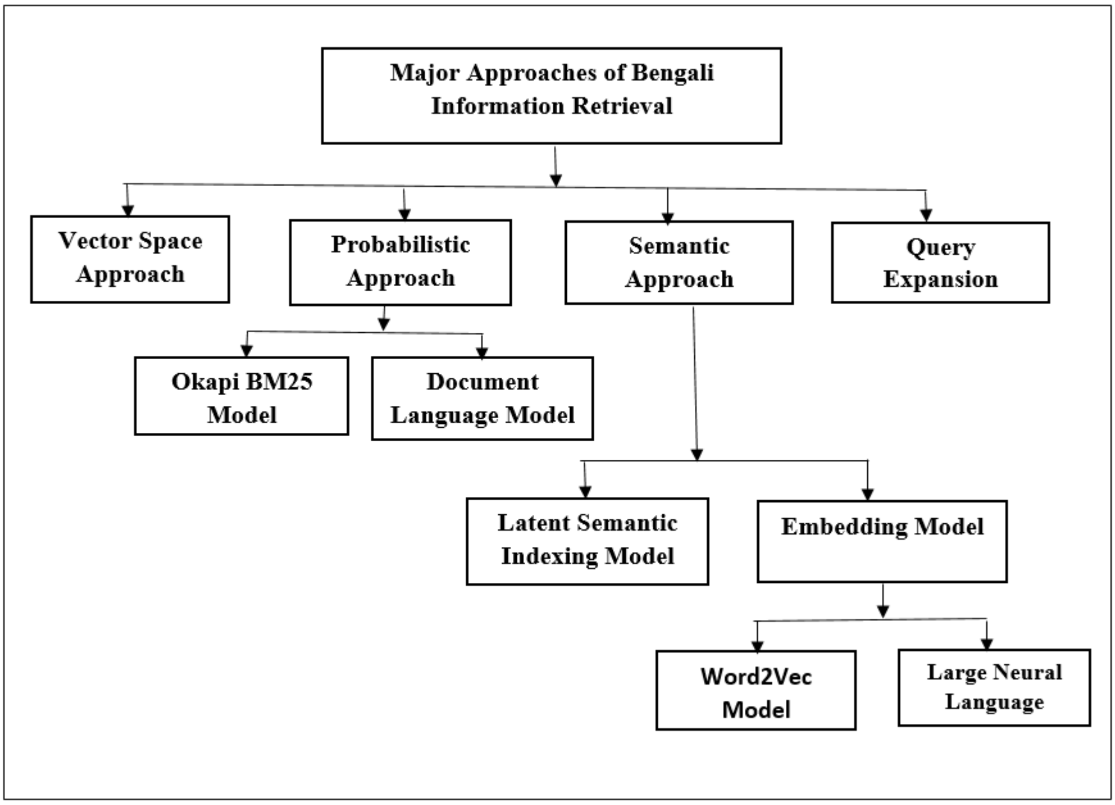


Figure 1.1: Major approaches to Bengali Information Retrieval.

1.4.1 Vector Space Model

In the vector space model(VSM), [10, 1], both the documents and queries are represented as vectors, and the relevancy and ranking of documents are determined based on the closeness between the corresponding vectors in the vector space.

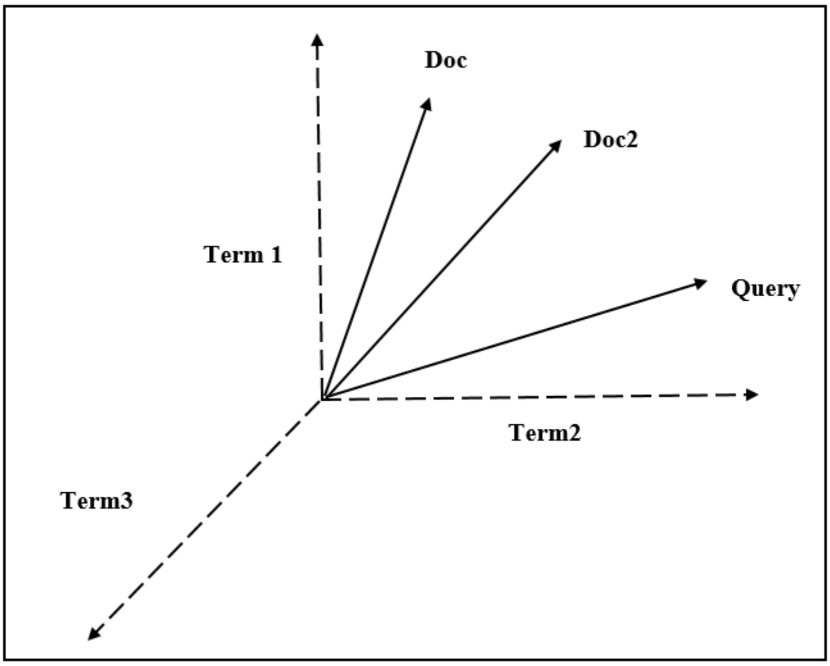


Figure 1.2: Vector Space Model.

VSM model captures the relative importance of the query terms present in the document. According to the VSM model, the documents and user query both are mapped to a high dimensional space called vector space. A component of a high-dimensional document vector corresponds to the TF*IDF weight of a vocabulary term if the term occurs in the document. If the term does not occur in the document, the corresponding vector component is set to 0. Similarly, a query vector whose length is equal to the vocabulary size is created. The TF*IDF weight of a term is calculated as the product of TF(Term Frequency) and IDF(Inverse Document Frequency) where TF is the frequency of a term in a given document and IDF(Inverse Document Frequency) is computed using the formula: log(N/df), where N is the number of documents in the corpus and df is the number of documents in which the term occurs at least once. However, after creating the query and the document vectors, documents are ranked based on the cosine similarity between the query vector and each document vector. The value of cosine similarity depends on the angle between two vectors, and so it ranges between 0 and 1. Here this similarity function is used as a ranking function that ranks documents for a given query.

The VSM model with TF*IDF heuristics has been one of the effective empirical information retrieval models. Several variants of this model have been proposed in the literature[1, 8, 9, 11], and many empirical studies have been carried out. Some researchers have applied the vector space(VS) Model for Bengali information retrieval.

Das and Mitra [12] present an approach for finding out the stems from a text in Bengali, which is highly inflectional in nature. In their process, they first stripped off the suffix part from Bengali words using some suffix stripping rules, depending upon the type of suffixes. Then they checked for the validity of the suffix-stripped word as a root word, using a Bengali dictionary. They have tested the process on Bengali collection of the FIRE 2010 data set with 50 queries using Lucene as the search engine and it gives a quite satisfactory result in terms of recall and MAP value. Lucene scoring uses a combination of the Vector Space Model (VSM) of Information Retrieval and the Boolean model to determine how relevant a given Document is to a User's query.

Kowsher et al. [13] utilized the VSM model to extract the correct answer from the document. They are determining the lemmatization of Bengali words using Trie and Dictionary Based Search by Removing Affix (DBSRA), as well as comparing the exact lemmatization with Edit Distance. They also created an Anaphora resolution method to ensure that information was expressed correctly. They also created a Bengali root word corpus, a synonym word corpus, a stop word corpus, and a corpus of documents.

Most researchers have used FIRE datasets[12, 12, 14, 15] for the implementation of Bengali IR models. We have tested our proposed models on FIRE datasets also.

1.4.2 Probabilistic approach

Unlike the Vector Space Model which is an algebraic model, the probabilistic model [16] works on the probability ranking principle and uses a probabilistic scoring function to retrieve documents. This model returns a ranked list of documents based on a probability score which is the estimated likelihood of relevance of a document to a query. It considers frequency statistics and the Bayesian theorem to find out the relevance score of documents for a user query. Many variants of the probabilistic model for IR have been proposed. The earliest probabilistic model is a Binary independence model[17]. As such we did not find any research on the Binary independence model in Bengali information retrieval. The most common probabilistic approaches are Okapi BM25 [18] and the language model.

Okapi BM25 approach

Okapi BM25 IR model, also known as BM25, [19, 18] as the basic IR model for retrieving the relevant documents from the collection of documents. This model[18] uses a scoring function that is almost similar to a TF-IDF-based vector space retrieval function. Additionally, Okapi BM25 combines the document-length normalization factor with TF*IDF heuristics for document ranking.

In a probabilistic IR system, queries are scored using a slightly different formula motivated by probability theory, rather than cosine similarity and tf-idf. Indeed, by simply adopting term weighting formulae from probabilistic models, people have been able to transform an existing vector-space IR system into a functionally probabilistic one. The following score function in BM25, as used by the Lemur Project [8], assigns a relevance score to document d with respect to the query q.

$$BM25\_SCORE(d,q) = \sum_{q_i \in q} IDF(q_i) \times TF\_factor(q_i,d) \tag{1.1}$$

where

$$TF\_factor((q_i,d)) = \frac{TF(q_i,d) \times (1+K1)}{TF(q_i,d) + K1 \times ((1-B) + B \times \frac{Doc\_Len(d)}{Avg\_Doc\_Len}))} \tag{1.2}$$

Where:
$TF(q_i,d)$ is the the number of times the word q_i occurs in the document d.
$Doc\_len(d)$ is the length of document d.
Avg_Doc_Len is the average document length in the corpus. The Inverse Document Frequency (IDF) of the word q_i is calculated in terms of DF as:

$$IDF(q_i) = log[0.5 + \frac{N}{DF(q_i)}] \tag{1.3}$$

Where $DF(q_i)$ is the document frequency that counts the number of documents in the collection C that contain the given word q_i.
N is the total number of documents in collection C.
$K1$ and B are the tuning parameters.

This score function is similar to the TFIDF score function because equation 1.1 has a TF part and IDF part[19]. In equation 1.1, d is represented as a document and q is represented as a query. Long documents have large TF values which have a dominating effect on a document's score. So, to normalize TF, a parameter $K1$ is used, to change TF to $TF/(TF+K1)$. The variable $K1$ is a positive tuning parameter that calibrates the frequency scaling of the document term. A $K1$ value of 0 indicates a binary model (no term frequency), while a large value indicates the use of raw term frequency.

The most basic premise is that, when two documents on the same topic are of different lengths, it is simply because the lengthier is more wordy. When viewed attentively from a linguistic standpoint, as representing a model of conversation, this is a fairly primitive assumption: it indicates that wordiness is due to repetition rather than greater elaboration, and so on. On this basis, the model interpretation should be expanded to normalize term frequency by document length. Another tuning parameter (b value belonging from 0 to 1) that determines scaling by document length is b : $b = 1$ is equivalent to fully scaling the term weight by the document length, whereas $b = 0$ indicates that no length normalization is performed.

One researcher Sarkar and Gupta [8] presented a comparative study on several IR models for the Bengali language (Bengali is an Indian language). In this study, the documents were stemmed using YASS stemmer [6], and TF*IDF-based vector space models and the Okapi BM25 model were implemented. Both models were evaluated using the FIRE dataset. This study revealed that the BM25 model performed better than the TF*IDF-based model.

Document Language modeling approach

Though BM25 has been proven to be one of the most effective retrieval models, another effective retrieval model is the language modeling approach to information retrieval. Unlike the VSM models and BM25 which rely on heuristic retrieval functions, the language modeling approaches use retrieval functions designed without much heuristics. The language modeling approach is mostly grounded on a probabilistic model that defines a probability distribution over a sequence of terms by which a document or a query is represented. Among the different types of language modeling approaches, the approach with the query likelihood retrieval function [20] has shown comparable or better performance than the traditional VSM model and BM25 [21]. Compared with the other IR models mentioned above, the language modeling approach to IR is more statistically sound and can optimize retrieval parameters by leveraging statistical estimation. It achieves comparable or better empirical performance without much parameter tuning. Ponte and Croft [20] successfully applied first a language modeling approach to IR. They used a query likelihood retrieval function that assigns a score to a document based on the probability that a given query is generated by the language model of the document.

Ponte and Croft [20] defined a ranking function that uses query likelihood scoring for the documents. They proposed a basic language modeling approach that builds a probabilistic language model M_D from each document D and assigns a score to D based on the probability of generating the query Q using the language model of document D, that is, it computes $Score(Q, D) = P(Q \mid D)$, which is called a query likelihood scoring function. The probabilistic language model M_D built from each document D is called the document language model because it views a document as a collection of words sampled from the author's mind and assumes that the high-frequency words are more important and the low-frequency words are arbitrary. The simplified form of the document language model assumes that the query terms occur independently given the document language model, and computes the probabilistic score $P(Q \mid M_D)$ for document D using the formula: $\prod_{w \in Q} P(w \mid M_D)$, where P$(w \mid M_D)$ =$\frac{tf(w,D)}{|D|}$, $tf(w, D)$= how many times w occurs in D and | D | = the number of terms in D. In this case, the probability is calculated based on maximum likelihood (ML) estimation which relies on the frequency of the word w in the document. Since the ranking score is calculated by the product of probabilities, the score becomes zero if one of them is zero. The probability can be zero if the corresponding query word does not occur in the document. Since a document is usually small in size, it is less likely that it contains all query words. Therefore the main problem with the ML estimator is that this leads to the zero probability problem because the probability P(w | M_D) becomes zero for the unseen words. This affects retrieval performance.

Ganguly and Mitra [22] implemented the LM framework in the Bengali language. Their experiments on TREC-8 and INEX 2007-08 collections have provided enough empirical evidence of the superiority of LM in terms of precision as well as usage simplicity as compared to VSM which requires tuning lots of parameters for high precision retrieval. They have proposed a simple rule-based stemmer for Bengali words to reduce the annotated words to their base form. They also suggest a very straightforward yet understandable query word importance for choosing the LM's initial parameters to increase precision.

Dolamic and Savoy [23] studied several IR models which are TF-IDF, Okapi BM25, Language Modeling, and various probabilistic models that used the Divergence from Randomness framework. For enhancing retrieval performance for the Hindi, Bengali, and Marathi languages, they created stopword lists and developed light-stemming strategies that remove inflectional affixes of nouns and adjectives [24]. In the query-document matching process, they used two types of frequency normalization techniques.

Paik et al.[25] proposed a novel graph-based language-independent stemming (GRAS) algorithm that is ideal for information retrieval. They test their method on seven languages with varied levels of morphological complexity utilizing collections from the TREC, CLEF, and FIRE evaluation platforms. The rule-based stemmers they used (GRAS) showed a substantial improvement. They used the TERRIER information retrieval system to conduct retrieval studies. Along with other well-known formulas like tf-idf and the language modeling approach, TERRIER implements a number of divergence-from-randomness-based weighting schemes. They weighted terms according to the IFB2 model [26] throughout their studies.

1.4.3 Semantic approach

The existing word-based vector space model [10, 8] suffers from the word mismatch problem when the query words and document words do not exactly match. Due to this problem, the traditional word-based vector space model gives poor recall. Stemming [7, 6] is usually used to improve the recall of IR systems. Moreover, though stemming can alleviate the word mismatch problem posed by inflectional word forms to some extent, it cannot handle the semantic level word match.

The main challenge of IR is how information can be effectively presented to the user with relevant information in response to a query. But one of the fundamental problems in information retrieval is the word mismatch problem which arises from the fact that the same question may be asked in different ways using different sets of words. It is also a fact that similar concepts may be presented in different ways in documents. A semantic information retrieval system attempts to generate search results based on context [27]. It automatically identifies the concepts structuring the texts. For instance, if you search for “passport” a semantic information retrieval system might retrieve documents containing the words “visa”, “embassy” and “flights”. What this means is that the search engine through natural language processing will know whether you are looking for a small animal or a Chinese zodiac sign when you search for “rabbit”.

The earliest approach to semantic-based IR follows latent semantic indexing(LSI) [28] for document representation. Recently, deep learning-based approaches for text representation have been popular Word2Vec model [29, 30] is an unsupervised deep learning method that represents each vocabulary word as a dense vector. The word embedding-based approach is another approach that is used for document representation or finding similarities between query and document words. BERT [31] is a pre-trained language model trained on a huge amount of corpus. BERT encoder is also used for representing documents or queries as vectors.

In this subjection, we will discuss these three methods and related existing research works on Bengali IR that used semantic methods for word, document, and query representation.

Latent Semantic Indexing (LSI) Model

Latent Semantic Indexing (LSI)(sometimes referred to as latent semantic analysis (LSA)) is an algebraic-statistical technique for representing meanings of words by their contextual usages and mapping documents into low-dimensional abstract concept space called latent semantic space where a concept is represented by the set of words appearing in similar contextual usages. In other words, it maps relations among terms and documents in semantic space. The rationale is that terms that occur in similar contexts will be positioned nearer to each other in the latent semantic space. The similarities between the documents and the queries in the latent semantic space indicate how they are semantically similar. LSI-based IR system attempts to find a match between documents and the queries based on concepts shared by them. The degree of relevance between documents and queries is then estimated by computing the cosine measure in the latent semantic space [32, 28].

Latent Semantic Analysis is done by applying the singular value decomposition (SVD) on a term-by-document matrix created using the entire corpus of documents. When SVD is applied to the term-by-document matrix, it produces three matrices which can be combined to produce a low-rank approximation to the original matrix. If r is the rank of the original matrix and C_k is the corresponding low-rank matrix with the rank k and k is far smaller than r, the dimensions associated with the contextually similar terms are combined (a combination of contextually similar words represented as an abstract concept). As a result, the documents are mapped to a k-dimensional concept space called latent semantic space. When the queries are mapped into the same space, the cosine similarity measure can be used to find conceptual overlap between a document and a query. In this case, if a document and a query share similar concepts will be mapped nearer to each other in the latent semantic space, and the cosine similarity value is considered as the relevance score[4]. At the time of testing the LSI-based model, the queries that are not part of the original matrix can be folded in by matrix multiplication [33]. The LSI-based model is useful[28, 34] in improving recall of the IR system.

There are limited research works on Bengali IR that use LSI-based document representation. Hoque et al.[35] used the LSI method for Bengali information retrieval. They tested their proposed model on a small Bengali corpus containing only 50 documents. They choose a similarity threshold value of 0.4, meaning that the document is considered relevant if the similarity value exceeds 0.4.

Das et al.[2] building an IR system named Anwesha, a prototype search engine for Bangla. They used diverse knowledge sources like WordNet, statistical co-occurrences (by way of Latent Semantic Analysis (LSA)), and external knowledge sources like Wikipedia. They also incorporated a lemmatizer into Anwesha. To evaluate their results, they also created their own dataset(a total of 1182 text documents) that contains query document relevance pairs of 907.

Although, for the above-mentioned two cases, variants of the LSI-based methods were used for Bengali IR, both methods were tested on a small dataset. They were not tested on the large benchmark dataset like the FIRE dataset used in our study.

Word embedding Based Approach

The LSI-based IR approach has some limitations. When the corpus size is large, it becomes difficult to apply LSI on a large corpus. On the other hand, the recent deep learning-based model is suitable for handling a large corpus of documents. Word2Vec model [29, 30] is an unsupervised deep learning model that produces word embedding which is used for finding semantic similarity between words that are represented as k-dimensional vectors in the vectors pace.

Word embedding(WE) is a popular and effective method [29, 36] for word representation and finding the semantic similarity between words. Technically, it represents a word as a vector and it maps the words which are contextually similar nearby in the embedding space. The concept of producing distributed representations is introduced here. They introduce some dependency of one word on the other words intuitively. This dependence would be larger for terms in the context of this word. The word embedding model represents a word as a vector rather than a string of characters. The embedding of the word vectors in k-dimensional space places the word vectors close to each other if they co-occur in similar contexts. Using the word embedding model, based on computing cosine similarities between the word vectors, we can easily identify a list of words that are used in similar contexts concerning a given word. Word embedding, in contrast to standard word representation, addresses challenges with data sparsity, high-dimensionality, and lexical gaps by capturing semantics and syntactic information in the form of dense vectors. Word embedding has drawn more attention in recent years and has successfully been applied to various NLP applications, including IR.

Word2vec model was first published in 2013 [29]. It is commonly used for producing word embedding from a large corpus of texts. The concept behind word2vec is that output is a word and input is a context, that is, the word's context. If two words occur frequently in the same context, they should be comparable, and we should acquire similar representations for them. There are two word2vec models: CBOW (continuous bag of words) and skip-gram. Both models are MLP (multi-layer perceptron) models with a d-dimensional input and a d-dimensional output, as well as $h < d$ hidden nodes in a hidden layer between them, and so resemble auto-encoders. Given a sentence and a center word, the input is the context and the output is the center word. The MLP is trained to predict the target word based on the words appearing in the context window. The context employs a window in which several words on both sides of the center are considered. In CBOW, the one-hot representations of all the words in the window are given as input. The input is a d-dimensional vector with non-zero values for all of the words in the context.

The skip-gram Word2vec model does the opposite of the CBOW model. The skip-bigram model predicts several context words based on a single input word. Its training instances are created by taking one word from the context at a time and pairing it with the center word. Thus, in the case of a skip-gram model, each input pair has only one nonzero element, but there will be multiple such pairs. Mikolav[29] concludes that "the Skip-gram model works slightly worse on capturing the syntactic relationship than the CBOW model, but it is better at capturing semantic relationships between words.

After the advent of the neural word embedding model, some researchers have applied it to solve the word mismatch problem faced by the Bengali IR system. Chatterjee and Sarkar [1] presented a Bengali information retrieval that used the word embedding-based document and query representation for computing similarity between a document and the query. In this work, the average of vectors of the words present in the document is used for document representation and the average of the query vectors is used for query representation. The similarity between a document and the query is computed using the cosine similarity measure.

Ganguly et. al. [37] word embedding for developing enhanced retrieval model. In their model, two possible cases are considered. A term is generated by a document or collection and then it is transformed into another term (a term in the query) after passing through a noisy channel. The transformation probability of the term in a noisy channel is calculated using the distances between the word vectors of these two terms.

Some researchers have used word embedding [38, 39, 40] for query expansion. In these approaches, the query is expanded by adding more terms selected globally from a corpus based on similarities between the query word vectors and corpus word vectors.

Large neural language model-based Approach

The major disadvantage of the Word2Vec model is that it produces just one vector representation for each word. As a result, if the word has multiple senses, they are combined into one single vector which is less context-dependent. This means a word has only one embedded vector in all the sentences in which it is used. But A word may appear in many sentences with different contexts. On the other hand, the BERT model produces multiple context-dependent embeddings for the same word depending on the context of the word.

FastText [41] is an open-source library for text representation developed by Facebook's AI Research. FastText is an improved word embedding model. In contrast to the Word2Vec model, FastText works at the character level. It breaks words down into smaller subword units, such as character n-grams for calculating word embedding. Thus, it can efficiently handle out-of-vocabulary problems. FastText can learn better representation for languages like Bengali which are morphologically rich. For training the FastText model, the CBOW (continuous bag of words) with negative sampling is used.

BERT(Bidirectional Encoder Representations from Transformers) [31], have recently been developed to simulate the underlying data distribution and learn linguistic patterns or characteristics in language. It can capture the meaning of a word in the right way. BERT is designed based on a Transformer that uses an attention mechanism that learn the contextual relationships between words (or subwords) in a text. Transformer's basic design consists of two independent mechanisms: an encoder that reads the text input and a decoder that does prediction. Since BERT acts as a language model, only the encoder part is used. The details of the transformer can be found in [42].

The Transformer encoder reads the entire sequence of words at once and processes the text input sequentially (from right to left or left to right). It is therefore thought of as bidirectional. Due to this feature, the model can determine the context of a word based on both its left and right surroundings.

BERT uses two training procedures: (1) Masked Language Modelling (MLM) and (2) Next Sentence Prediction. In the MLM method, some percentage of words in the input sequence are masked, that is, those words in each sequence are replaced with a [MASK] token. The model is then trained to predict the original value of the masked words, based on the surrounding words of the masked word. In the second training method known as next sentence prediction (NSP), the model receives pairs of sentences as input and learns to anticipate whether the second sentence in a pair will come after the first sentence in the original text. Training input pairs are created from the documents in such a way that half of the input sentence pairs in which the second sentence is the subsequent sentence of the first sentence in the original text, and in the remaining half of sentence pairs, the second sentence is a sentence randomly chosen from the corpus.

Although BERT is pre-trained using a large volume of unlabelled data, it can be fine-tuned for a specific task. The pre-trained BERT model may be fine-tuned with just one additional output layer to generate a model for a wide range of tasks, such as question answering and language inference, without requiring significant task-specific architecture changes. As a result, even search workloads with limited training data can benefit from the pre-trained model. On numerous NLP tasks, this model outperformed standard word embeddings [43, 31, 44]

There are many advantages of the BERT model over the Word2Vec model or the FastText Model. The BERT model processes the input sequence as a whole to produce contextual embedding. It considers WordPieces (subwords) while generating embedding. Hence the out-of-vocabulary problem is not a crucial issue when the BERT embedding is used. Moreover, the BERT model explicitly takes into account the position of each word in the sentence for computing the word embedding.

Various pre-trained BERT models are available. In general, there are two types of BERT models-BERT_bASE and BERT_large. The difference between these two models is the number of encoder layers. BERT_base model has 12 stacked encoder layers whereas BERT_large has 24 layers of stacked encoders. They also include feed-forward networks with 768 and 1024 hidden units, respectively, and 12 and 16 attention heads, respectively.

Although BERT has been applied for various NLP tasks, we found limited research on Bengali information retrieval using BERT. Das et al. [45] used two-stage ranking-based document retrieval. In the first stage, the traditional TF-IDF-based retrieval was used. In the second stage, BERT-based contextual sentence embeddings were used to rerank the candidate documents retrieved in the first stage. For re-ranking, each document was represented by the average embeddings of the document's top sentences that are most relevant to the query. They created a gold-standard Bengali dataset containing query document relevance pairs.

1.4.4 Query Expansion-based Approaches

We have discussed the traditional IR models in the earlier subsections. Out of the various IR models adopted by the researchers for the Bengali language IR, two influential IR models are the vector space model[10] and the statistical language model[20]. The main drawback of this kind of IR model is that they ignore the relevant documents if there is no query term common between the documents and the given query. This problem is known as the word mismatch problem, which occurs when the same topic is posed in various ways. Furthermore, it is true that related ideas can be presented in many ways in various texts.

Word mismatch problem is a crucial problem for an IR system. Many researchers have suggested popular solutions in order to overcome the word mismatch problem. Most early studies on Bengali information retrieval [7, 4, 6] have used the term normalization techniques, stemming, or lemmatization for solving the word mismatch problem in Bengali IR. They mainly focused on developing a Bengali stemmer for stemming both documents and queries and incorporating it into the traditional IR models like Vector Space Models, probabilistic models, Okapi BM25 IR models, or language models. Most researchers used YASS stemmer[6] to stem the queries and documents and improve the efficacy of an IR model. The most common dataset used by early researchers for Bengali IR evaluation is the FIRE dataset.

Ganguly et al. [37] proposed a novel morphological technique for de-compounding Bengali words and applied it to improving Bengali IR. They used the frequency of the constitu-ents of a compound word to break it into multiple parts. They claimed that this method was useful in enhancing word frequency in the corpus and hence it improved the IR performance.

Apart from the morphology-based method, query expansion (QE) is another method that is very effective in overcoming word mismatch problems[46]. This method involves choosing terms or phrases through local or global analysis, adding them to a user's initial query, and forming an expanded query which is used to refine the search. It reduces mismatch between query and document and hence improves IR performance. Broadly, there are two types of query expansion strategies:(1) global analysis and (2) local analysis. In the global analysis, QE approaches implicitly choose words for reformulating the initial query from global knowledge resources or large corpora[47, 48]. Many researchers [49, 50, 51, 52] have used Wikipedia as a knowledge source for extracting expansion terms.

In contrast with the global analysis method, the local analysis-based QE techniques choose expansion terms from the document collection returned by the user's first query. Here the assumption is that some of the documents returned by the initial query are relevant; thus, words found in the returned documents should be relevant to the initial question as well. The local analysis methods can be of two types- Relevance feedback(RF) and Pseudo-relevance feedback(PRF). The RF approach allows users to provide feedback after returning an initial set of search results in response to a user query and utilizes this feedback information for a better representation of the query. Rocchio is one of the earliest relevance feedback approaches used in the traditional vector space framework and it generates a modified query vector which is obtained by computing first the weighted sum of the original query vector and the centroid of the document-set judged as relevant, and then subtracting the weighted centroid of the non-relevant set from the sum [53]. Salton and Buckley [54] used a query re-weighting method that is influenced by Rocchio's method [53] for relevance feedback and its subsequent advances. However, the relevance feedback approach is not popular because the users are reluctant to provide relevant feedback. Pseudo-relevant feedback(PRF) [55, 56]approach enriches the user query according to the highest-ranking documents returned in response to the initial query. PRF has been proven to be an effective query expansion mechanism [4]. For the Indian languages, PRF has also been used for developing an effective IR model. Atreya et.al. [57] have proposed a multilingual query expansion framework in several Indian languages including Bengali. They enhanced the PRF model with a query expansion technique that combines the expansion terms selected from the highest-ranked k documents in the query language(resource-scarce language) with the expansion terms selected from the highest-ranked k documents in an assistive language (a resource-rich language). Prasath and Sarkar[58] used the CBD

(Clustering-by-Directions) algorithm proposed by Kaczmarek(2011) [59] to find additional terms from the documents retrieved by the initial query and add the extracted terms to the query for expansion. Ganguly et al. [60] presented two methods for expanding queries in Bengali information retrieval (IR).

Word embedding(WE) is a popular and effective method [29, 36] for word representation and effective in finding the semantic word similarity. Technically, it represents a word as a vector and it maps the words which are contextually similar in close proximity in the embedding space. Word embed-ding, in contrast to standard word representation, addresses challenges with data sparsity, high-dimensionality, and lexical gaps by capturing syntactic and seman-tics information in the form of dense vectors. The usefulness of word embedding for query expansion has been demonstrated by various researchers in [61, 38]. Chatterjee and Sarkar [1] presented a Bengali information retrieval that combines the TF-IDF-based VSM model (Vector Space Model) with another IR model that uses the word embed-ding-based document and query representation for calculating similarity between a document and the query.

The fusion of multiple expansion techniques has shown useful in extracting more informative query expansion terms. Some researchers [62, 63, 64, 65, 66] suggested a hybrid method for query expansion that combines statistical and semantic methods for term selection. Wang and Niu [65] introduced a hybrid query expansion method that enhances query expansion performance by combining global analysis and ontology technology. They considered co-occurrences of terms calculated using the global analysis method and semantic reasoning using ontology. Zingla et al. [66] combined statistical, semantic, and conceptual approaches to generate new related terms that were used for expanding a given query. Very recently, Sharma et al. [64]proposed an IR model for retrieving information from medical documents. They expanded the user queries with the medical terms selected based on the term scores where a score of a term is calculated by combining the WordNet-based semantic score and the BERT Score for the term.

1.5 Major Approaches to Summarizing Search Results

Search engines provide aid in managing information overload problems and assist users in finding relevant information. Although search engines do aid in the reduction of information overload, they create a new problem by returning too many web pages for a single query. As a result, the user frequently needs to scan through hundreds of pages to find which documents are useful. Because of this, research has revealed that users of search engines frequently quit after looking at the first several documents. To help these users more effectively explore the retrieval set, it would be very beneficial if an efficient search engine could be created to help group the retrieved web documents into clusters and provide summary information. To improve the users' search experience, the IR system can be integrated with document clustering multi-document summarization modules. In this subsection, we will present a survey on the summarization of search results.

Since the documents returned by the conventional IR system are clustered, a summary of each cluster is produced. Text summarization is a process that produces a condensed version of a document or a document set. Considering the input size, the text summarization system can be of two types: single document summarization (SDS) system [67] which creates a summary from only one document and a multi-document summarization (MDS) system [68] creates a single summary from a cluster of documents. The summary produced by either an SDS system or an MDS system can be of two types: extractive summary[69, 70, 71, 69, 72, 73] and abstractive summary [74, 75, 76]. The extractive summary is composed of salient sentences or sentence segments selected only from the input. In this thesis, my focus is on extractive multi-document text summarization because our target is to create a summary from the clusters of documents returned by a conventional IR system.

Most existing unsupervised extractive multi-document text summarization systems calculate the score of a sentence using word importance and it is further combined with sentence position-based score, similarity to the title, etc. [77]. The earliest work that identifies the important words as keywords for text summarization is Luhn's work [78] which identified the keywords based on the number of occurrences of the word in the input text. Then a sentence is given more importance if it contains more keywords. But the method of measuring sentence importance based on the number of keywords assigns equal importance to multiple sentences if they contain the same number of keywords. To overcome this limitation, most summarization methods assign TF*IDF weights to words[79, 80, 81, 82]. The TF*IDF method assigns a weight (importance) to a word by computing the product of TF and IDF, where TF is the frequency of the term (word) in the input and IDF is the inverse document frequency computed using a background corpus. In the TF*IDF method, TF is multiplied by IDF to obtain TF*IDF weight for each term, which is used for measuring term importance.IDF is computed using the formula: log(N/DF), where N is the total number of documents in the corpus and DF is the number of documents in which the word appears.

Some researchers have estimated word weights using supervised approaches and features such as probability and location of word occurrence [83, 84, 85]. The hybrid method proposed in [86] combines machine learning algorithms and statistical methods for identifying the important words (keywords) and calculates sentence scores based on the number of keywords the sentence contains.

Many recent works also calculate the word importance using semantic relations between terms. Sarkar and Dam[68] also generate a multi-document summarization using the word importance which is computed based on traditional term frequency and semantic term relations. In this work, word embedding-based similarity between terms is used to compute the degree of semantic relations.

Ravinuthala and Chinnam[87] introduced a graph-based approach to keyword extraction and calculating sentence scores based on the summation of the weights of keywords contained in the sentence. Sarkar[88, 67] proposed a novel work that identifies candidate concepts from a document, ranks them based on their weights, and assigns a score to a sentence based on the most important concepts it contains. Finally, a subset of sentences is chosen to create a summary. There have been only limited research works that use a machine learning algorithm for predicting word importance. The earliest works [89, 90, 91] on machine learning-based text summarization focused on predicting the summary-worthiness of a whole sentence using a set of sentence-specific features. To our knowledge, there is only one work[92] that uses a machine learning algorithm for predicting keywords and utilizes the keywords in the text summarization process.

Although text summarization has evolved as a separate research field. There are a limited number of research works that focus on the clustering of search results and generating summaries from the clusters. Radev and Fan [3] focused on open-domain multi-document summarization within a Web search framework. They designed a prototype system that integrates classical information retrieval technology with advanced document clustering and multi-document summarization technology. Their system supports Boolean search and Vector Space search. A user has the option of selecting a search method. A user can also combine Boolean and Vector Space searches. They used centroid-based sentence extraction for summarization. They conclude that such a retrieval system aids in improving reading speed and determining the relevancy of the retrieved web pages.

Bidulya and Spryiskov proposed a [93] unique multi-document summarising approach for search results. As a result of the first phase's search request, the system obtains the document set. The sentences with the highest weights are extracted from each document and used to create a summary in the second stage. The weight assigned to each sentence in a given document depends on how they relate to one another. The ”raw” abstract of the document consists of these extracted sentences. In the third stage, the system generates a summary of the search results from ”raw” abstracts. They used a modified TF-IDF measure to determine the documents' keywords.

Pasupathi et al.[94] proposed a novel idea of creating a comparison summary from the search results. From the search engine's returned results, the user selects a collection of links to web pages. Comparative summaries for these selected websites are created. This approach makes use of the HTML/XML DOM tree structure of these websites. In HTML documents, various concept blocks are split into parts. The sentence score of each concept block is determined by the query and feature keywords. Sentence weight is determined by taking into consideration factors such as the quantity of query string occurrences, feature keywords, and their proximity and frequency, the position of the sentence, the tag in which the text appears in the document, uppercase words, etc. To quickly produce a comparison summary, the important phrases from the concept blocks of different websites are extracted. This method saves the consumer time and effort from surfing various websites to compare the information. Users would gain from making decisions more quickly.

Li et al [95] suggested a new method for summarising multimedia news for Internet search results that finds underlying topics in query-related news data and produces a query-related summary. They presented the hierarchical latent Dirichlet allocation (hLDA) model to identify one representative news article for each topic after discovering the hierarchical topic structure from query-related news documents, followed by a new approach that used weighted aggregation and max pooling. Each topic is illustrated with one representative image in addition to the text content. To provide a concise and cohesive overview of each topic's parent topic, they suggested using the time-bias maximum spanning tree (MST) technique to weave together the representative documents for each topic. To give users a hierarchical summary of the news information they need, they then build a user-friendly interface. They demonstrate the effectiveness of the suggested news summarising technique for news retrieval.

1.6 Evaluation Metrics

Since our thesis is on developing an effective IR system and summarizing the search results returned by the IR system. We have evaluated the IR component and summarization component separately. In this section, we discuss IR evaluation metrics and summarization evaluation metrics.

1.6.1 IR evaluation metrics

Mean Average Precision(MAP) is a well-known measure used to evaluate the retrieval model's effectiveness [96]. In this thesis, we have also used MAP for evaluating the proposed IR models.

To compute MAP, we need the ranked list of documents generated by a retrieval model in response to a given query and the relevance file containing the human relevance judgments for the given query. The relevance file gives us information that helps to check whether a retrieved document is relevant to the query or not. The average precision (AP) is calculated using the following formula.

$$AP(q_i) = \frac{1}{m_i}\sum_{r=1}^{n} P(r) \tag{1.4}$$

Where: r is the relevant document's position in the ranked list.
n is the number of documents retrieved by the IR model
$\mathrm{m_i}$ refers to the total number of relevant documents for the query q_i
$P(r)$ is precision at position r.
The formula used to determine precision at position r.

$$P(r) = \frac{rel_r}{r} \tag{1.5}$$

Where: rel_r is the total number of relevant documents that were retrieved up to position r.

The mean of average precision (MAP) is calculated considering average precision (AP) for all queries as follows.

$$MAP(Q) = \frac{1}{|Q|} \sum_{q_i \in Q} AP(q_i) \tag{1.6}$$

1.6.2 Summary Evaluation Metrics

Summary evaluation is a subjective task because, for a given document or a document set, the reference summaries produced by different human judges may not be exactly the same. Since human evaluation for a system-generated summary is subjective, automatic summary evaluation tools are available for evaluating summaries. The popular automatic evaluation package is ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [97].

ROUGE package evaluates system summaries by comparing each system summary with a set of reference (model) summaries and reports evaluation scores in terms of ROUGE-N scores which are computed by counting word N-grams common between a system summary and the human summaries (reference summaries).

$$ROUGE\_N = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in N} Count_{match}(gram_n)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in N} Count(gram_n)} \tag{1.7}$$

Where n stands for the length of the n-gram,
$gram_n$, and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries.

1.7 Scope of the Thesis

In this thesis, we undertake the problem of Bengali text document retrieval, clustering retrieved documents, and summarization of the document clusters. A summary of the scope of the thesis:

- Develop novel techniques to solve IR problems.
- Explore the traditional language modeling-based approach and develop a novel technique for enhancing the smoothing process.
- Explore both the traditional IR models, incorporation of deep text representation in the IR models, and hybridization of IR models where various novel query expansion techniques are combined.
- Explore combining multiple IR models for effective retrieval and develop novel techniques for model combination.

- Cluster the documents retrieved by the IR model and produce an effective summary of each cluster.

- Develop a novel technique for clustering and summarization of search results.

- Evaluation of the IR and summarization models using benchmark datasets.

1.8 Motivation and Contribution of the Thesis

Nowadays non-English online users are rapidly increasing on the Internet Google-KPMG estimated that 536 million Indian users would use regional languages on the Internet by 2021. (The article was published in the Economic Times on April 30, 2021). The major search engines are trying to add support for the Indian languages including Bengali.

Compared to English, the Bengali language has a more complex morphology. Bengali is a language in which verbs and nouns have many inflected forms. In the Bengali language, proper nouns can appear in various forms. Compared to nouns, verbs are more inflected in Bengali. Most earlier studies focused on improving Bengali IR models using morphological analysis like stemming and lemmatizing, but a limited number of attempts have been made by researchers to improve Bengali information retrieval models using lexical, semantic, and/or morphological information. The earlier studies did not also focus on combining multiple IR models to improve retrieval performance.

The above-stated reasons motivate us to explore the challenges of information retrieval in a low-resource language like Bengali.

A large number of documents are usually retrieved by an IR model in response to a query. It is not an easy task for a user to go through each article and find relevant information quickly. Clustering of retrieved documents and displaying summaries of the produced clusters to the users helping them to spot the relevant cluster and quickly find the relevant information by unfolding the cluster. This gives us additional motivation to design and develop a system that can summarize search results returned by an IR model.

The main contributions of the thesis are as follows:

- In Chapter 2, we propose to improve the document language model for Bengali information retrieval, we have integrated a hybrid smoothing method with the document language model. The proposed hybrid smoothing method combines our new cluster-based smoothing method with a collection-based smoothing method. Since the cluster-based smoothing method relies on the quality of the clusters created from a corpus of documents, we have used a word embedding-based incremental clustering algorithm to produce better clustering results.

- In Chapter 3, we explore the semantic approaches for Bengali information retrieval. To overcome the word mismatch problem and query document matching problem, we use the latent semantic indexing model, a Word embedding-based model for Bengali information retrieval. We also use BERT embedding to obtain a better document vector and this is incorporated into the Bengali IR model.

- In chapter 4, we propose a hybrid query expansion approach. We combine statistical, lexical, and word embedding-based methods to develop a hybrid query expansion method that is effective for Bengali information retrieval.

- In chapter 5, we explore a model combination method for Bengali information retrieval. The traditional word-based vector space model captures the relative importance of query words present in the documents and the word embedding-based model or the LSI model captures the semantic level similarity between query words and document words to overcome the word mismatch problem. Therefore, we hypothesize that the outputs of the two types of models can be combined using proper blending functions that can improve information retrieval performance.

- In chapter 6, we explore the clustering of documents retrieved by an IR model and summarization of each cluster. We introduce several novel features for measuring word importance(weight) used for developing a multi-document summarization system that accepts a cluster of related documents and produces a single summary. For multi-document summarization, we calculate the sentence score by summing up the weights of the important words contained in the sentence where a support vector regressor is trained for predicting word weights. The total score of a sentence is calculated by combining this score with the positional score. Finally, sentences are ranked based on their combined scores and the top k non-redundant sentences are selected to form an extractive summary of a document cluster.

1.9 Organization of the Thesis

We organize our thesis as follows:

- In Chapter 2, we present a hybrid smoothing method with the document language model for information.

- In Chapter 3, we focus on the effects of semantic-based approaches in Bengali information retrieval to overcome the word mismatch problem.

- In Chapter 4, we present a novel hybrid query expansion strategy in this chapter. We assess these models on benchmark datasets. The three expansion term extraction techniques—using WordNet for the first extraction, word embedding for the second, and term frequency and the four hybrid models this chapter presents for the third—are combined to create three models that account for all potential combinations.

- In Chapter 5, we focus on the effects of the hybrid model on IR. We compare existing IR systems and propose a hybrid IR model to overcome the word mismatch problem. We investigate the effectiveness of the hybrid model used to improve Bengali information retrieval.

- In Chapter 6, we present a multi-document summarization (MDS) approach that computes the score of each sentence based on word importance predicted by a support vector regression (SVR) model and produces a multi-document summary by ranking sentences according to sentence scores.

- In Chapter 7, we examine the contributions of our research and discuss possible future research avenues to wrap up the thesis.

2

Cluster-based Smoothing for Language Model-based Bengali Information Retrieval

2.1 Introduction

Over the years, many different scoring functions have been proposed for developing different IR models. Two such IR models are the Vector Space Model (VSM) [10] and the Probabilistic model [98, 99, 19, 8]. The Vector Space Model (VSM) [10] represents each document as a vector. This model captures the relative importance of the terms present in the document. According to the VSM model, the documents and user query both are mapped to a high dimensional space called vector space. A component of a high-dimensional document vector corresponds to the TF*IDF weight of a vocabulary term if the term occurs in the document. If the term does not occur in the document, the corresponding vector component is set to 0. Similarly, a query vector whose length is equal to the vocabulary size is created. The TF*IDF weight of a term is the product of TF (Term Frequency) and IDF(Inverse Document Frequency) where TF is the frequency of a term in a given document and IDF (Inverse Document Frequency) is calculated using the formula: $log(N/df)$, where N is the number of documents in the corpus and df is the number of documents in which the term occurs at least once. However, after creating the query vector and the document vectors, documents are ranked based on the cosine similarity between the query vector and each document vector. The value of cosine similarity depends on the angle between two vectors, and so it ranges between 0 and 1. Here this similarity function is used as a ranking function that ranks documents for a given query. The VSM model with TF*IDF heuristics has been one of the effective empirical information retrieval models.

The above-stated IR models were heavily used in the English language domain. For the Bengali language, IR research is limited. Most researchers who worked on Bengali IR have proposed stemming or lemmatization methods for improving Bengali retrieval performance [6, 7]. Some researchers have tested the performance of the BM25 model on the Bengali IR dataset [8]. The language modeling approach is less experimented with in the Bengali IR domain. A few attempts [23, 25] have been made to develop a language modeling approach to Bengali information retrieval. The performance of the language modeling approach to IR heavily depends on the smoothing technique used to handle the zero-probability problem which occurs when the query word does not occur in the document. When a document is usually small in size, it is less likely that it contains all query words. Recently, word embedding has been popular for many natural language processing tasks. FastText [100] is a word embedding model that learns term representations by using sub-word information. The FastText word embedding model captures fine-level more granular information and the true relationships between texts. For this reason, it overcomes out-of-vocabulary problems to a major extent. FastText word embeddings are now available for 157 languages including Bengali. Since Bengali is a highly inflectional language, the out-of-vocabulary problem is more severe for Bengali information retrieval. We hypothesize that incorporating an improved smoothing method in the language modeling approach and the use of FastText embedding in the smoothing process resolves the out-of-vocabulary problem to some extent and improves the IR performance.

In this chapter, we propose a cluster-based smoothing method for improving a language modeling-based approach to Bengali information retrieval. The proposed smoothing method uses cluster information where documents are clustered using a histogram-based clustering method with FastText word embedding-based document representation. For implementing the language modeling approach, we have followed the work proposed by Ponte and Croft [20] who built a document language model based on only document-level estimates and computed the probability of a query given a document language model, that is, P(Q|MD) where Q is the query and MD is the document language model. On the other hand, for the smoothing method, we have enhanced the smoothing method proposed by Liu and Croft [101]. We combine cluster-based smoothing with a collection-based smoothing method. Since the performance of cluster-based smoothing depends on clustering quality, we improve smoothing by enhancing the clustering algorithm. Finally, we have developed a Bengali information retrieval system that uses a language model that is smoothed by the proposed cluster-based smoothing method combined with a collection-based smoothing method. In our work, the cluster-based smoothing method has been enhanced by improving document clustering quality using a histogram-based incremental clustering technique[102] and a semantic similarity function that uses FastText word embedding-based document representation.

In the first part of this chapter, we discuss the query preprocessing method and histogram-based incremental clustering technique to cluster the relevant documents. Then we describe the proposed clustering-based smoothing method and finally, we integrate the clustering-based smoothing method with the document language model for ranking documents.

In the second part of this chapter, we describe the benchmark datasets on which the proposed model is tested. The proposed Bengali IR system has been tested on two datasets, the first one is the FIRE dataset which is a well-known benchmark dataset for Bengali information retrieval and the second one is a gold standard dataset which is developed by Das et al.[2]. Experiments on the benchmark datasets show that our proposed technique achieves state-of-the-art performance for Bengali information retrieval. The evaluation shows that the proposed clustering-based smoothing technique can produce improved performance.

The rest of the chapter is organized as follows. In section 2.4.3, we present an error analysis to show where the proposed technique fails and what are the possible reasons for failures. Finally, we conclude in Section 2.5. In this section, we also suggest some future work for further enhancement of the proposed technique.

2.2 Proposed Methodology

Our proposed work is an enhancement of the work by Liu and Croft [101]. They used a language modeling approach to IR which was improved by various smoothing techniques. They applied two smoothing methods for smoothing the maximum likelihood estimates given by a document language model (the document language model is discussed later in this section). Two smoothing methods used by them were (1) the Jelinik-Mercer smoothing method, and (2) a hybrid smoothing method that combines a collection-based smoothing method with a cluster-based smoothing method.

Our proposed approach is based on the document language model which is smoothed using a hybrid smoothing method that combines our proposed cluster-based smoothing method with a collection-based smoothing method. Since the cluster-based smoothing method depends on the quality of clusters produced from the corpus of documents, we have used a histogram-based incremental clustering technique [102] and a semantic similarity function for producing better clustering results and enhancing the cluster-based smoothing method using the information drawn from the obtained clusters. After enhancing the cluster-based smoothing method, it is combined with the collection-based smoothing method to achieve a better smoothing effect. Liu and Croft [101] used a bag-of-words document representation and the K-means clustering algorithm for document clustering, whereas we have used a histogram-based clustering algorithm that relies on word embedding-based document representation and a semantic similarity function. Therefore, our proposed IR model for Bengali information retrieval is based on a language modeling approach that uses an improved smoothing method for achieving a better smoothing effect on the maximum likelihood estimates given by the document language model.

Our proposed IR system for the Bengali language has four major steps which are as follows.

1. Query Preprocessing
2. Document clustering using a word embedding-based incremental clustering algorithm[102] and a semantic similarity function
3. Building the document language model and smoothing it using an improved hybrid smoothing method

4. Ranking documents based on the smoothed language model and the given query

The architecture of our proposed IR system is shown in Figure 2.1. We discuss separately each step of the proposed IR model in this section.

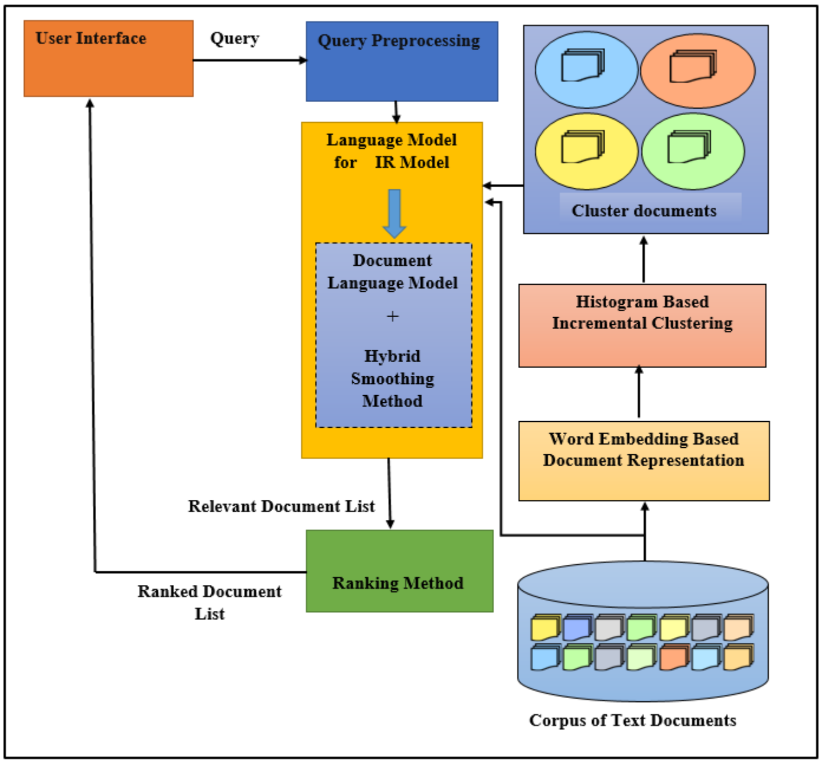


Figure 2.1: Architecture of our proposed model

2.2.1 Query Preprocessing

The input to the proposed IR model is a query that is processed first. Query preprocessing involves tokenization, stemming, punctuation removal, and stop-word removal. Stop-words were removed using the list of stop-words provided by FIRE. Finally, the queries are then tokenized into a collection of words. Each query is stemmed using the Bengali stemmer named Yet Another Suffix Stripper (YASS) [6].

2.2.2 Document Clustering

Our proposed approach to Bengali IR is based on a document language model which is smoothed using an improved cluster-based smoothing method combined with a collection-based smoothing method. Since the cluster-based smoothing method relies on the information drawn from a set of clusters produced from a corpus of documents, we need to apply a document clustering algorithm for producing a set of clusters. We have used word embedding for document representation and a histogram-based incremental clustering technique [102] for producing better clustering results. In this subsection, we discuss first how documents are represented using word embedding and then we discuss how the histogram-based clustering algorithm is used for document clustering.

Word embedding-based document representation

Document representation plays an important role in clustering documents. If we represent any document properly then it can improve clustering performance. For this reason, we represent a document not only using the syntactic method, but we represent it using a semantic method. For this purpose, we have used a word embedding-based method for document representation. Word embedding represents each word by k-dimensional vectors in a vector space. The words that occur frequently in a similar context and have very near meanings are assigned vectors that are closely positioned in the k-dimensional vector space. The document-representation process using word embedding is discussed below. We have used the pre-trained model which was developed by Grave et al. [100] for the Bengali language to get the vector for each word. This pre-trained model is trained on Common Crawl and Wikipedia data. The dimension of each word vector is 300. Each document is represented by selecting keywords from the document. We obtain the document vector by averaging the vectors of the selected keywords. We did not use all words in the document for document representation because noisy and unimportant words may result in producing distantly positioned document vectors for conceptually similar documents and vice versa. For keyword selection, the words are assigned scores which are computed using Equation (2.1). The score of a word w (stop words are not considered) is calculated as follows:

$$Score(w) = \frac{1}{2} \times (\frac{TF(w) \times IDF(w)}{Max(TF(w) \times IDF(w))} \times \frac{1}{\sqrt{i}}) \tag{2.1}$$

Where: i is the position of a word in a document.

*Max TF(w)*IDF(w)* = Maximum of the TF-IDF values for the words that occurred in the corresponding document.

TF(w): Term frequency is the number of times the word w occurs in the document

DF(w): Document frequency, DF is computed by counting the number of documents containing the word w in the corpus.

IDF(w): Inverse document frequency, IDF is computed over a corpus of N documents using Equation (2.2).

$$IDF(w) = log[0.5 + \frac{N}{DF(w)}] \tag{2.2}$$

After ranking the words based on scores computed using Equation (2.1), the top-ranked m words are chosen and the document vector is produced by taking the average of the vectors for the selected m keywords. We have used cosine similarity measures to find similarities between two document vectors.

Incremental clustering

After representing each document using word embedding, we adopted a histogram-based incremental clustering algorithm introduced by Hammouda and Kamel [102] for creating a set of clusters from the entire collection of the documents. The clusters produced by this clustering algorithm are fed to our proposed language modeling-based IR model. The information drawn from the obtained clusters is used for smoothing maximum likelihood estimates given by a document language model. Therefore, clustering is an important step of our proposed IR model.

The histogram-based incremental clustering algorithm produces clusters in a single pass by maximizing the tightness of clusters based on the similarity distribution inside each cluster. Similarity distribution inside each cluster is monitored by computing pairwise similarities among the documents within the cluster and computing the histogram of similarities. For this clustering technique, the similarity measure plays an important role in introducing more tight clusters. The primary modification that we have made on this incremental clustering method is to incorporate our defined semantic similarity measure while computing pairwise document similarities and producing clusters of high quality. We have used cosine similarity measures to compute the similarity between two document vectors corresponding to a pair of documents. Document vectors are the dense vectors obtained using the method discussed in the earlier sub-section. The important feature of this clustering technique is that the number of clusters is automatically inferred and it produces cohesive clusters in a single pass. This is why we have chosen this clustering method for improving our proposed Bengali IR system.

The working strategy of the incremental clustering algorithms is to process each data object (document vector) at a time and incrementally assign this data object to the appropriate cluster using some criteria defined based on the histogram representing similarity distribution inside each cluster. Here the crucial decisions are: how to assign the next object to the appropriate cluster? How to maintain cluster cohesiveness? The histogram-based clustering algorithm is designed to add a data object to an appropriate cluster by maintaining cluster cohesiveness as much as possible. The cluster cohesiveness is represented by the similarity histogram of each cluster. The similarity histogram of a cluster is a statistical representation of similarity distribution calculated using pairwise similarities among the documents within a cluster. To create a similarity histogram for each cluster, we select the number of bins based on the similarity interval. We divide the range of similarity values from 0 to 1 into ten equal intervals. Each bin height is determined by the count of pairwise document similarities that fall within the similarity interval for the respective bin. Figure 2.2 shows a similarity-histogram of clusters. The ideal cluster has a histogram where all the similarities are grouped under the right-most bins.

A similarity histogram helps to monitor cluster cohesiveness. Before adding a document to a cluster, the clustering algorithm judges whether adding the document to the cluster can improve cluster cohesiveness or not. If it is improved, the document is added to the cluster, otherwise, it is not added. In some situations, this strong strategy may create a problem. When a document has a high similarity to most of the documents but still it may be rejected by the cluster because of a much stricter policy. So a little degradation (pre-specified by a tolerance range) is allowed when adding a document to the cluster. If it is observed that no cluster satisfies the criterion, a new cluster is formed with the document. To incorporate this idea in the clustering algorithm, we need to formalize what we have discussed above. Most importantly, we need to define criteria for measuring cluster cohesiveness. Cluster cohesiveness is measured by histogram ratio which is computed by the ratio of the count of similarities above the predefined similarity threshold (set to 0.8 in our case) to the total count of similarities in a cluster. The cluster is much more coherent when this ratio is very high.

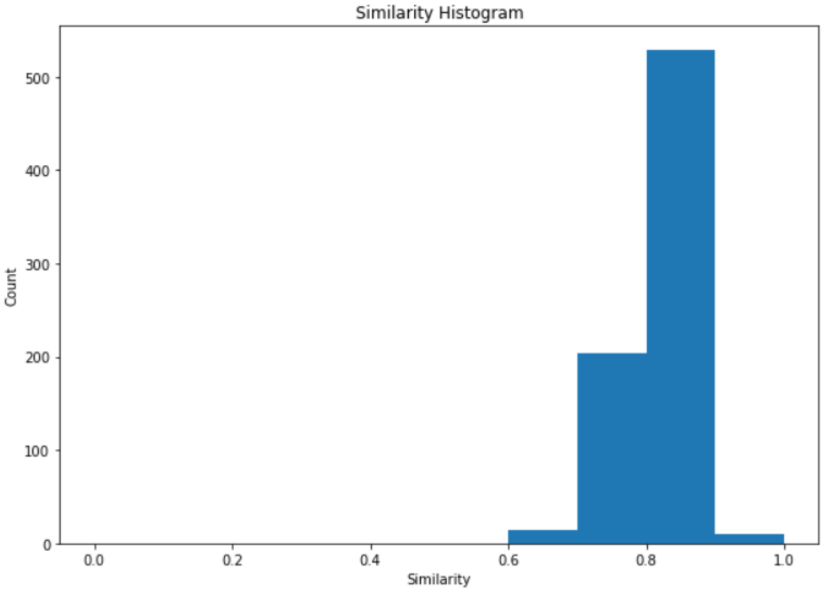


Figure 2.2: Cluster Similarity Histogram

Suppose N_{cl} is the total number of documents in a cluster. While simulating adding a document to a cluster, the number of pairwise document-to-document similarities for the cluster is M_{cl}= N_{cl} (N_{cl} + 1) / 2. Let S_{cl}=S_r: r=1,2… M_{cl} is a set of pairwise document similarities in a cluster. The similarity histogram of the cluster is represented by:

$$Histogram\ \ similarities = histo_b : b = 1, 2, ..Bin \tag{2.3}$$

$$histo_b = count(s_c) \qquad s_{lb} < s_c < s_{ub} \tag{2.4}$$

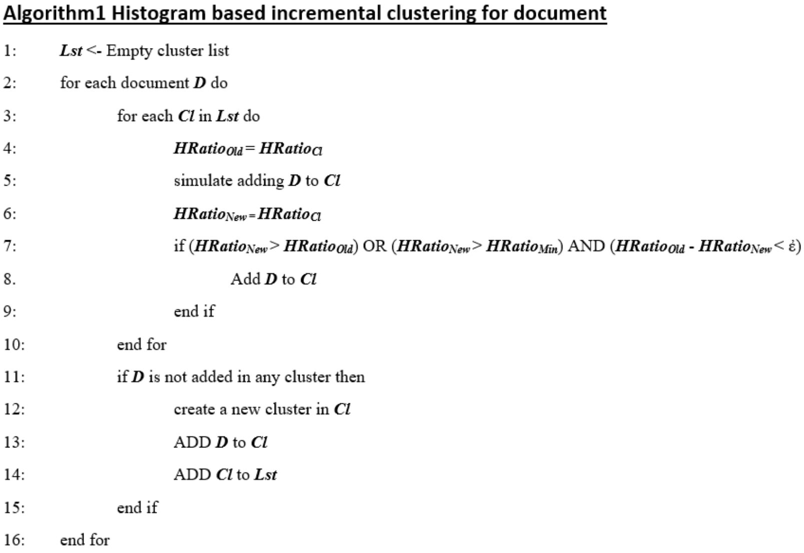
Where: Bin: the number of histogram bins
$histo_b$: the count of similarities in bin b
s_{lb}: the lower similarity bound of bin b, and
s_{ub}: the upper similarity bound of bin b.

The cluster cohesiveness is measured by histogram ratio (HRatio) for cluster cl and it is calculated as:

$$HRatio_c l = \frac{\sum_{b=TH}^{Bin} histo}{\sum_{j=1}^{Bin} histo_j} \tag{2.5}$$

$$V = \lfloor S_{TH} \dot{B}in \rfloor \tag{2.6}$$

Where: $HRatio_c l$: the histogram ratio of cluster cl
S_{TH}: the similarity threshold (0.8 in our case)
TH: the bin number corresponding to the similarity threshold.



Algorithm1 Histogram based incremental clustering for document

```
1:   $Lst$ <- Empty cluster list
2:   for each document $D$ do
3:       for each $Cl$ in $Lst$ do
4:           $HRatio_{Old} = HRatio_{Cl}$
5:           simulate adding $D$ to $Cl$
6:           $HRatio_{New} = HRatio_{Cl}$
7:           if ($HRatio_{New} > HRatio_{Old}$) OR ($HRatio_{New} > HRatio_{Min}$) AND ($HRatio_{Old}$ - $HRatio_{New} < \dot{\varepsilon}$)
8.               Add $D$ to $Cl$
9:           end if
10:      end for
11:      if $D$ is not added in any cluster then
12:          create a new cluster in $Cl$
13:          ADD $D$ to $Cl$
14:          ADD $Cl$ to $Lst$
15:      end if
16:  end for
```

Figure 2.3: Algorithm of Histogram-based incremental clustering for documents

Figure 2.3 shows the incremental clustering algorithm. In line 7 of the algorithm, the criterion for adding a document to a cluster has been mentioned. According to this criterion, if the new histogram ratio obtained after the simulation of adding the document to a cluster is greater than the old histogram ratio of the cluster or the difference between the new and old histogram ratio is within a predefined tolerance value (ε which is set to 0.1 for best results) and the new histogram ratio is greater than a predefined threshold ($HRatio_{min}$ which is set to 0.7 in our case), the document is added to the respective cluster. After checking all clusters, if a document is not added to any cluster, a new cluster is formed with this document. Thus this algorithm maintains cluster cohesiveness while assigning a document to a cluster.

2.2.3 Document Language Model and Smoothing Methods

The language model is the central part of the proposed Bengali IR system. We have used the document language model introduced by Ponte and Croft [20] for further enhancement using an improved smoothing method. In this section, first, we discuss the document language model in detail and then we discuss two smoothing methods which are (1) the Jelinik-Mercer smoothing method, (2) a hybrid smoothing method that includes our proposed cluster-based smoothing method.

Document Language Model

An interesting class of probabilistic models called language modeling approaches to information retrieval gives us effective scoring functions without using many heuristics. Due to their good empirical performance and great potential for leveraging statistical estimation methods, the language modeling approaches have received much attention since Ponte and Croft's pioneering work [20] that was published in 1998. Ponte and Croft proposed a query-likelihood retrieval model that ranks documents based on how likely it is that the query is generated using the corresponding language model. They built a document language model based on only document-level estimates and computed *P(Q | M_D)* where *Q* is the query and M_D is the document language model. The document language model views a document as a collection of words in which certain words have higher probabilities than other words. Thus it considers that each document has a language model and assigns a score to document *D* based on the probability that the given query *Q* is generated using the language model corresponding to document *D*. In other words, in this approach, documents are ranked based on the probability that they are on the same topic as the query. But the most common approach is to consider each query as a group of independent words and compute query likelihood by the product of individual query word probabilities [101], that is, the relevance score for document *D* and the given query, *Q* is computed as follows.

$$P(Q \mid D) = \prod_{w \in Q} P(w \mid D), \tag{2.7}$$

Where *w* is a query term in the query and *P(w | D)* is usually computed using maximum likelihood (ML) estimation as follows.

$$P(w \mid D) = \frac{tf(w, D)}{\mid D \mid}. \tag{2.8}$$

Where *tf(w,D)* = how many times *w* occurs in *D* and | *D* | = the number of terms in *D*.

One of the problems with the above-mentioned document language model is that it computes the maximum likelihood estimates for the words based on their frequency in the document under consideration. Since an individual document is usually small in size, many query words are not present in the document. In this situation, the maximum likelihood estimator assigns zero probability to the query words that are not present in the document. The zero probability problem occurs when the probability *P(Q | D)* (see Equation (2.7)) becomes zero for the unseen word w contained in a query. Since the documents are usually small in size, the zero probability problem is a critical issue that needs to be addressed to improve IR performance. So, our target is to use an improved smoothing method for smoothing the maximum likelihood estimates obtained from the document language model. In the next subsection, we discuss several smoothing methods in detail.

Smoothing Methods

In our language modeling approach to IR, the document language model has been enhanced using an improved hybrid smoothing method. In this subsection, we discuss several smoothing methods along with our proposed cluster-based smoothing method.

Jelinik-Mercer Smoothing Method

This smoothing method is a widely used smoothing method for dealing with the zero probability problem that affects the performance of a language modeling-based IR system. The objective of this smoothing technique is to assign non-zero probabilities to the unseen words. The Jelinik-Mercer Smoothing method smooths the maximum likelihood estimate, $P(w \mid D)$ obtained from the document language model using an interpolation technique given in Equation (2.9).

$$P(w \mid D) = \alpha P_{ML}(w \mid D) + (1 - \alpha)P_{ML}(w \mid Corpus) \tag{2.9}$$

Where: $P_{ML}(w \mid D)$ is the maximum likelihood estimate of the word w in the document (document language model),
$P_{ML}(w \mid Corpus)$ is the maximum likelihood estimate of the word w obtained from the corpus (collection language model)
α is the smoothing parameter.

Equation (2.9) shows how the maximum likelihood estimate obtained from the document language model is smoothed using a collection language model. The collection language model computes the maximum likelihood estimate $P_{ML}(w \mid Corpus)$ using Equation (2.10).

$$P_{ML}(w \mid Corpus) = \frac{TF(w, Corpus)}{\sum_{w' \in V} TF(w', Corpus)} \tag{2.10}$$

Where $TF(w, Corpus)$ = Total number of times the word w occurs in the corpus and V is the size of the vocabulary.

When the maximum likelihood estimates are smoothed using a collection language model, we call it the collection-based smoothing method which is used very often for dealing with the zero probability problem. As we can see from Equation (2.9), even if a query word does not occur in the document, that is, $P_{ML}(w \mid D)$ becomes zero for the word w, some probability mass is assigned to the word by the collection language model.

The limitation of the Jelinik-Mercer Smoothing method is that, in this method, smoothing is done using the same collection and all the unseen words in different documents would have similar probabilities.

Hybridizing Our Proposed Cluster-Based Smoothing Method with the Collection-Based Smoothing Method

To overcome the limitation of the Jelinik-Mercer smoothing method, the document language model can be smoothed by a hybrid smoothing method. Liu and Croft [101] used a hybrid smoothing method in which a document language model is smoothed by a cluster-based smoothing method and a collection-based smoothing method. The cluster-based smoothing method is based on a cluster-based language model[101]which computes the maximum likelihood estimate for each word w in a document (i.e., P($Q \mid Cluster$)) using the frequency statistics of the cluster containing the document under consideration. The cluster-based language model computes a relevance score for a document based on how likely a query Q is generated using the language model for the cluster containing the document, i.e. it computes P($Q \mid Cluster$) which is estimated as follows.

$$\prod_{w \in Q} P(w \mid cluster), \tag{2.11}$$

Where $P(w \mid cluster)$ is computed based on the frequency of the word w in a given cluster using Equation (2.11).

$$P(w \mid cluster) = \frac{TF(w, Cluster)}{\sum_{w \in cluster} TF(w, Cluster)} \tag{2.12}$$

Where TF(w, Cluster) = Total number of times the word w occurs in the cluster. In a cluster-based smoothing method[101], the maximum likelihood estimates given by the document language model are smoothed as follows.

$$P(w \mid D) = \alpha P_{ML}(w \mid D) + (1 - \alpha) P_{ML}(w \mid Cluster) \tag{2.13}$$

While combining the collection-based smoothing method with the clustering-based smoothing method, it is done in two steps. In the first step, the cluster-based language model is smoothed using a collection-based language model, and in the second step, the document language model is smoothed using the smoothed cluster-based language model. The cluster-based language model is smoothed using a collection-based language model as follows.

$$\begin{aligned} P(w \mid D) = \alpha P_{ML}(w \mid Cluster) + (1 - \alpha) P_{ML}(w \mid Corpus) \\ = \alpha \frac{TF(w, Cluster)}{\sum_{w \in cluster} TF(w, Cluster)} + \\ (1 - \alpha) \frac{TF(w, Corpus)}{\sum_{w' \in V} TF(w', Corpus)} \end{aligned} \tag{2.14}$$

Where:
$TF(w, Cluster)$ = Total number of times the word w occurs in the cluster
$TF(w, Corpus)$ = Total number of times the word w occurs in the corpus
V is the size of the vocabulary

After obtaining the smoothed cluster-based language model using Equation (2.14), it is used for smoothing the document language model. Equation (2.15) shows the two-step hybrid smoothing process.

$$P(w \mid D) = \alpha P_{ML}(w \mid D) + (1 - \alpha)[\beta P_{ML}(w \mid Cluster) + (1 - \beta) P_{ML}(w \mid Corpus)] \tag{2.15}$$

Equation (2.15) is obtained by combining the Equations (2.13) and (2.14). Equation (2.15) is our smoothed final language model. Effectively, the final model is a document language model smoothed by two smoothing methods, the cluster-based smoothing method and the collection-based smoothing method, which are applied in a certain order. Here the α and β are smoothing parameters. The smoothing method which is shown in Equation (2.15) interpolates three language models, namely, the document language model, the cluster-based language model, and the collection language model.

We hypothesize that the efficacy of the cluster-based smoothing method depends on the quality of the clusters produced from the corpus of documents. Our main target is to improve cluster-based smoothing using a suitable clustering algorithm. Liu and Croft[101] employed the K-means clustering algorithm for creating clusters of similar documents whereas we have used word embeddings-based document representation and a histogram-based incremental clustering technique presented in the previous subsection. Therefore, our main focus is to produce quality clusters so that a better estimation of P($W \mid D$) can be done through smoothing.

2.2.4 Ranking documents

As mentioned earlier in this section, we use the query likelihood ranking function for ranking documents with respect to a given query. For a given query Q and a document D, the query likelihood ranking score is computed using Equation (2.7), but, in our proposed model, P(w | D) from Equation (2.7) is replaced by the right-hand side of Equation (2.15), that is, we rank documents based on Equation (2.16) which is derived by combining Equation (2.7) and Equation (2.15).

$$P(Q \mid D) = \prod_{w \in Q} P(Q \mid D) = \prod w \in Q[\alpha P_{ML}(w \mid D) + \\ (1-\alpha)[\beta P_{ML}(w \mid Cluster) + (1-\beta) P_{ML}(w \mid Corpus)]] \quad (2.16)$$

Where: α and β are smoothing parameters that are empirically determined.

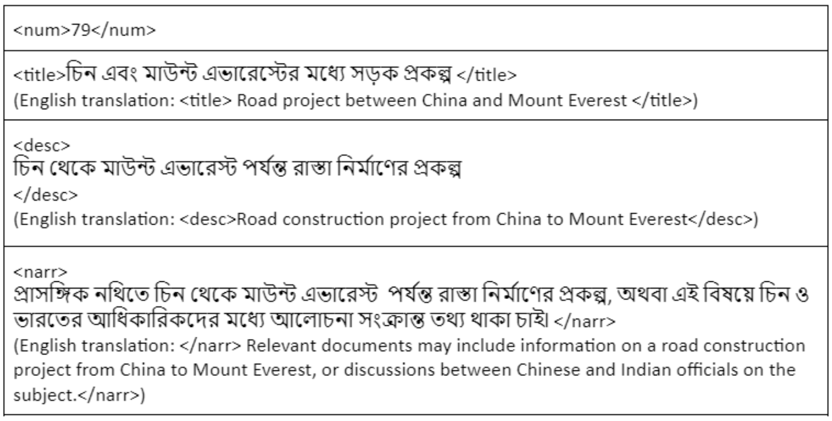
For efficient implementation of our Bengali IR system, the language models are developed at the beginning and loaded at index time. Finally, after assigning a relevance score to each document, a list of ranked documents is produced as the output of the IR model.

2.3 Description of the Data Set

We have tested the proposed IR model on two different datasets- 1) FIRE(Forum for Information Retrieval Evaluation) 2010 benchmark dataset which is mentioned as Dataset-1 in the rest of the paper, and 2) another Bangla dataset developed by Das et al. [2], we mention this dataset as Dataset-2 in the rest of the paper.

2.3.1 Dataset-1

We have used the FIRE(Forum for Information Retrieval Evaluation) 2010 dataset for our experiments. This dataset contains 123021 documents and a relevant file. The relevance file contains 50 queries(query numbers 76 to 125) and the human relevance judgment for each query. Each row of the file contains (Query-number, document-id) pairs where query-id is the number of the concerned query and document-id is the ID of the document for which relevance is judged with respect to the query. The relevance is indicated by binary relevance score : 0 or 1, where relevance = 1 if the corresponding document is relevant to the query and relevance = 0 if it is not relevant to the query. All documents are encoded in UTF-8. The dataset contains mostly news articles collected from the famous daily Bengali newspaper Ananda Bazar Patrika. Each query in the dataset is specified by a title, narration, and description as shown in Figure 2.4.



<num>79</num>

<title>চিন এবং মাউন্ট এভারেস্টের মধ্যে সড়ক প্রকল্প </title>
(English translation: <title> Road project between China and Mount Everest </title>)

<desc>
চিন থেকে মাউন্ট এভারেস্ট পর্যন্ত রাস্তা নির্মাণের প্রকল্প
</desc>
(English translation: <desc>Road construction project from China to Mount Everest</desc>)

<narr>
প্রাসঙ্গিক নথিতে চিন থেকে মাউন্ট এভারেস্ট পর্যন্ত রাস্তা নির্মাণের প্রকল্প, অথবা এই বিষয়ে চিন ও ভারতের আধিকারিকদের মধ্যে আলোচনা সংক্রান্ত তথ্য থাকা চাই। </narr>
(English translation: </narr> Relevant documents may include information on a road construction project from China to Mount Everest, or discussions between Chinese and Indian officials on the subject.</narr>)

Figure 2.4: A sample Bengali query (Topic Number 79) taken from FIRE 2010

2.3.2 Dataset-2

This is a gold standard Bengali IR dataset recently developed by Das et al.[2] for evaluating some basic IR models. This dataset contains a total of 1182 text documents out of which 182 documents are taken from short stories, novels, and essays written by Rabindranath Tagore and 1000 newspaper articles were crawled from the Bangla newspaper Prothom Alo in 2013. The collection contains news articles of the ten different categories: বাংলাদেশ/Bānlādēśa(EN: ‘Bangladesh’), খেলা /khēlā (EN: ‘sports’), বিজ্ঞান ও প্রযুক্তি/bijñāna ō prayukti (EN: ‘technology’), বিনোদন/binōdana (EN: ‘entertainment’), আন্তর্জাতিক/ āntarjātika (EN: ‘international’), অর্থনীতি// arthanīti (EN: ‘economy’), জীবনযাপন/ jībanayāpana (EN: ‘life-style’), মতামত// matāmata (EN: ‘opinion’), শিক্ষা/ śikṣā (EN: ‘education’) and আমরা/ āmarā (EN:‘we-are’). This dataset has 94 queries with different complexity levels-
The number of queries at complexity levels 1 and 2 is 26
The number of queries at complexity level 3 is 19
The number of queries at complexity level 4 is 23.

2.4 Evaluation, Experiments, and Results

To evaluate the proposed retrieval models, all the IR models have been tested on Dataset-1 and Dataset-2, as described in the earlier section. As we have mentioned earlier, in all the experiments, both the queries and documents are stemmed, and stop words are removed.

2.4.1 Evaluation

Mean Average Precision(MAP) is a well-known measure used to evaluate the retrieval model’s effectiveness [96]. To compute MAP, we need the ranked list of documents generated by a retrieval model in response to a given query and the relevance file containing the human relevance judgments for the given query. The relevance file gives us information that helps to check whether a retrieved document is actually relevant to the query or not. The average precision(AP) is calculated using the following formula.

$$AP(q_i) = \frac{1}{m_i}\sum_{r=1}^{n} P(r) \tag{2.17}$$

Where: r is a relevant document's position in the ranked list.
n is the number of documents retrieved by the IR model
m_{i} refers to the total number of relevant documents for the query q_i
$P(r)$ is precision at position r. The formula below is used to determine precision at position r.

$$P(r) = \frac{rel_r}{r} \tag{2.18}$$

Where: rel_{r} is the total number of relevant documents that were retrieved up to position r. The mean of average precision (MAP) is calculated considering AP for all queries as follows.

$$MAP(Q) = \frac{1}{|Q|}\sum_{q_i \in Q} AP(q_i) \tag{2.19}$$

2.4.2 Experiments

We have conducted the following experiments to prove the effectiveness of the Bengali IR system that uses a language modeling approach combined with our proposed cluster-based smoothing method.

Experiment 1

We have implemented our proposed language modeling approach to Bengali information retrieval that uses word embeddings-based document representation and incremental clustering for producing document clusters which are used for smoothing the document language model using Equation (2.15). We have developed two variants of our proposed IR model. They differ in how documents are represented. Our first model selects the top m keywords (m =150 for our case) from each document and calculates a document vector by averaging word vectors for the selected keywords. The detailed procedure for keyword selection and obtaining document vectors has been presented in the methodology section. The second model computes a document vector by taking the average of the vectors for all words in the document. The second model is compared with the first model to justify the effectiveness of keyword-based document presentation.

Experiment 2

In this experiment, we implement the BM25 model presented by Spärck Jones et. al. [19]. This is existing work in the literature and we consider it as the baseline model to which our proposed model is compared. The details of the BM25 can be found in [19]. For all queries, the parameters for the BM25 model are tuned through this experiment, and finally, the model is configured as $K1$ is set to 2.2, B is set to 0.2, where $K1$ and B are the important parameters of the BM25 model.

Experiment 3

In this experiment, we implement the cluster-based language model for IR presented by Liu and Croft [101]. This is an existing work in literature. This model is the closest to our model because it also uses Equation (2.15) for computing P(w|D), but our work differs from Liu and Croft's work in the type of clustering algorithm used for producing clusters provided to the cluster-based smoothing method. Liu and Croft's work [101] used the bag-of-words model for document representation and the k-means clustering algorithm whereas we have used word embeddings-based document representation and the histogram-based incremental clustering algorithm for producing better clustering results which are used for improving the cluster-based smoothing method. We have considered the model proposed by Liu and Croft [101] as a baseline model and it is implemented for comparison with our proposed model. For the implementation of Liu and Croft's model, we have set the parameters as mentioned in [101].

2.4.3 Results and Discussion

Table 2.1 shows the MAP scores obtained by the variants of our proposed two IR models for Dataset-1 and Dataset2. The variants of our proposed model are (1) the proposed model with document representation using selected keywords, and (2) the proposed model with document representation using all words. The results shown in Table 2.1 reveal that our proposed model with keyword-based document representation performs better than the model that represents a document using all words. This also justifies the fact that the noisy words present in the document hamper clustering performance and the poor clustering results affect the retrieval performance. This establishes the fact that the cluster-based smoothing method depends on clustering quality. As we can also see from the table, the proposed model performs better on both datasets it is tested on. This also shows the generalization capability of the proposed model.

Comparison with existing methods

We have chosen three existing models for comparison with our proposed best model. The first model is the widely used IR model called the BM25 model [19]. This model is based

Table 2.1: Performance Comparisons of our proposed two models based on MAP score for Dataset-1 refers to FIRE 2010 dataset and Dataset-2 developed by other researchers Das et al.[2].

Model Name	Dataset-1	Dataset-2
	MAP	MAP
`Our proposed model with document representation using selected keywords`	0.5157	0.4467
`Our proposed model with document representation using all words`	0.5039	0.4326

on the probabilistic relevance framework and it is proven to be effective for IR in various domains. The second model is the basic document language model smoothed using the Jelinik-Mercer Smoothing method (Equation (2.9), and the third model is the cluster-based document model proposed by Liu and Croft [101]. The cluster-based document model presented in [101] applied a hybrid smoothing method that combines a cluster-based smoothing method with the collection-based smoothing method for smoothing $P(w \mid D)$. This model used the bag-of-words model for document representation and the k-means clustering algorithm for producing clusters which are used for smoothing P(w|D) using Equation (2.15). Although Liu and Croft's work [101] is closest to our work, we have used a different clustering algorithm for producing better clustering results which are used for smoothing $P(w \mid D)$ using Equation (2.15). Instead of using the K-means clustering algorithm and the bag-of-words document representation, we have used a histogram-based document clustering algorithm and word embedding-based document representation to produce better clustering output that results in a better smoothing effect on the document language model. A performance comparison of our proposed best model with the other three baseline models is shown in Table 2.2. The results show that our proposed IR model performs significantly better than the baseline models it is compared to. We can also see from this table that the proposed model performs better than the baseline models on Dataset-2 as well. This proves the generalization capability of the proposed model.

We have calculated the percentage improvement in the performance of our proposed approach over the baseline models using Equation (2.20) and we present percentage improvement results in Table 2.3.

$$Performance\ Improvement(PI) = \frac{y-z}{z} \times 100 \tag{2.20}$$

Where:
y=MAP Score obtained by the Proposed approach
z=MAP Score obtained by baseline approach

Table 2.3 shows the best performance improvement achieved by the proposed IR model over the three baseline models, namely, the cluster-based language model proposed by Liu

Table 2.2: Performance comparisons of our proposed best models with other three baseline models using Dataset1 and Dataset-2

Model Name	Dataset-1	Dataset-2
	MAP	MAP
Our proposed model with document representation using selected keywords	0.5157	0.4467
The cluster-based language model proposed by Liu and Croft [101]	0.4917	0.4295
BM25 model [19]	0.4863	0.4289
The basic document language model with Jelinik-Mercer smoothing (Equation (2.9))	0.4790	0.4132

Table 2.3: Percentage improvement achieved by our proposed best model over the baseline models using Dataset-1 and Dataset-2

Model Name	Dataset-1	Dataset-2
	MAP	MAP
The cluster-based language model proposed by Liu and Croft [101]	4.88	4.004
BM25 model [19]	6.05	4.15
The basic document language model with Jelinik-Mercer smoothing (Equation (2.9))	7.66	7.19

and Croft [101], the BM25 model [19], and the document language model with Jelinik-Mercer smoothing (Equation (2.9)). Performance improvements over these three baseline models for Dataset-1 are respectively 4.88%, 6.05%, and 7.66%. On the other hand, performance improvements over the baseline models for Dataset-2 are respectively 4.004%, 4.15%, and 7.19%. For both datasets, more than 4% improvement achieved by the proposed model over the baseline language model proposed by Liu and Croft [101] indicates that the proposed clustering-based smoothing method helps in improving the overall performance of the language model-based Bengali information retrieval system. We also observed that our proposed model performs significantly better than the well-known IR model named BM25.

Parameter tuning

In this subsection, we present how we obtain the optimal values for the model parameters. We observed that the most important parameter whose values affect the retrieval performance of our proposed model is m which is the number of keywords used for document representation. Figure 2.5 shows the impact on our proposed model's retrieval performance on Dataset-1 when m is varied. The figure shows that the best results are obtained when m is set to 150.

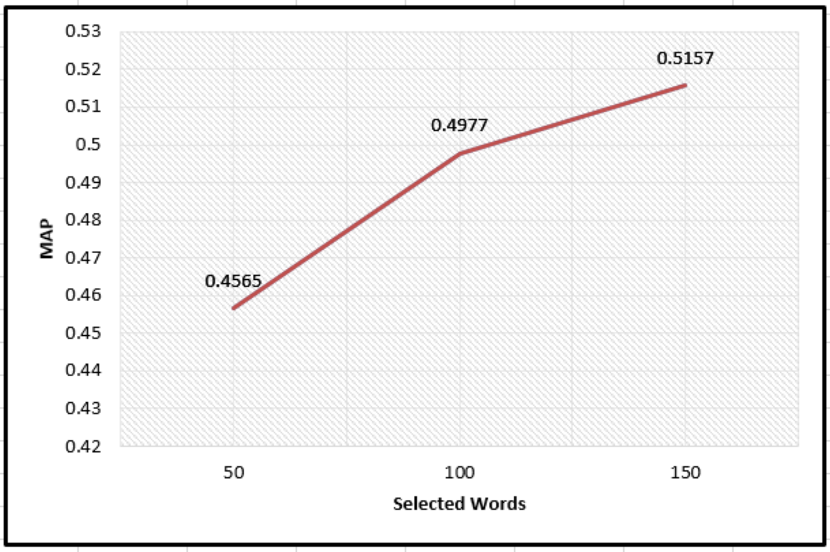


Figure 2.5: Impact on our proposed model's retrieval performance on Dataset-1 when m is varied

Since we have used Equation (2.15) for smoothing query probability, it has two important tuning parameters α and β which need to be tuned for better retrieval performance. A sequential search has been performed to find the optimal values for these parameters. Both the parameters have been varied between 0 and 1 sequentially. We have shown in Figure 2.6 the impact on our proposed model's retrieval performance on Dataset-1 when the parameter α is varied between 0 and 1 but β is fixed to 0.5.

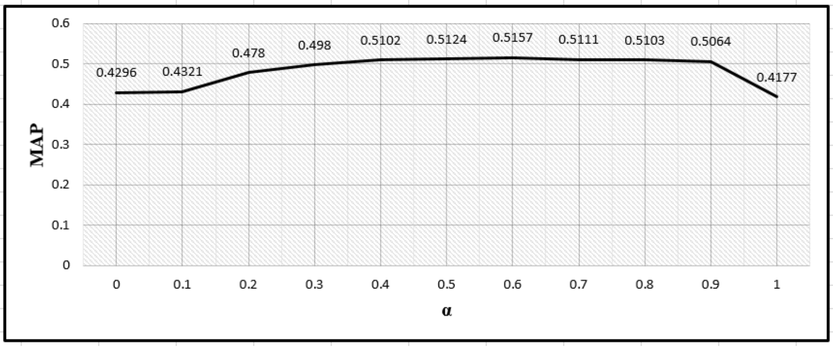


Figure 2.6: Impact of smoothing parameter α on our proposed IR model's retrieval performance on Dataset-1 when α is varied and β is fixed to 0.5

As we can see from Figure 2.6, the best retrieval performance in terms of MAP score is obtained when α is set to 0.6

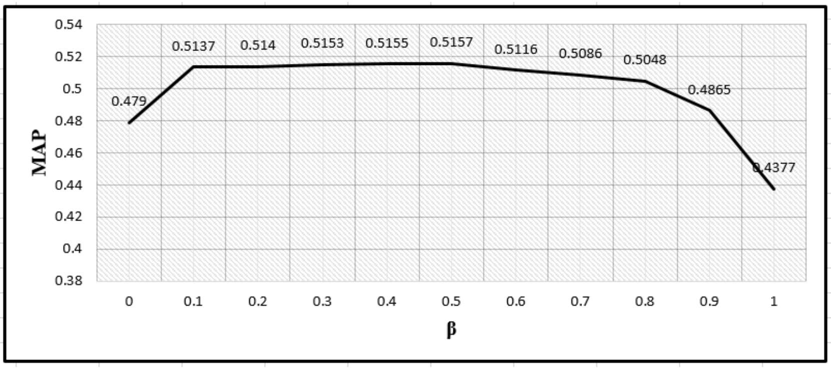


Figure 2.7: Impact of smoothing parameter β on our proposed IR model's retrieval performance when β is varied, but α is fixed to 0.6

By setting α to its optimal value of 0.6, the smoothing parameter β is varied between 0 and 1. We have shown in Figure 2.7 the impact on our proposed model's retrieval performance when β is varied It is observed from Figures 2.6 and 2.7 that the best results on our dataset are obtained when the values of the smoothing parameters α and β are set to 0.6 and 0.5 respectively.

For Dataset-2, the model parameters m, α, and β are set to the values for which we obtain the best results on Dataset-1.

Impact of Document Length and Query Size on Retrieval Performance

We have conducted several experiments to judge the impact of document length and query length on retrieval performance. The first experiment is done to find whether there exists any correlation between query length and retrieval performance achieved by the proposed model. Figure 2.8 and Figure 2.9 show the plots of query length vs MAP scores for Dataset-1 and Dataset-2, respectively. We can see from these figures that there exists no correlation between the query length and MAP scores.

The second experiment is conducted to find out whether the length of a document has any relation with the retrieval performance. To do this, we study the impact of document length on retrieval performance. To examine the impact of document length on retrieval performance, we plot rank position vs. average document length. For example, the average document length is computed by taking the average of the lengths of documents ranked at position 1 for all queries in a dataset. Figure 2.10 and Figure 2.11 illustrate the correlation between the rank position and the average document length for Dataset-1 and Dataset-2, respectively. By analyzing the figures, we cannot conclude that the longer documents are ranked earlier in the position than the shorter ones or vice versa.

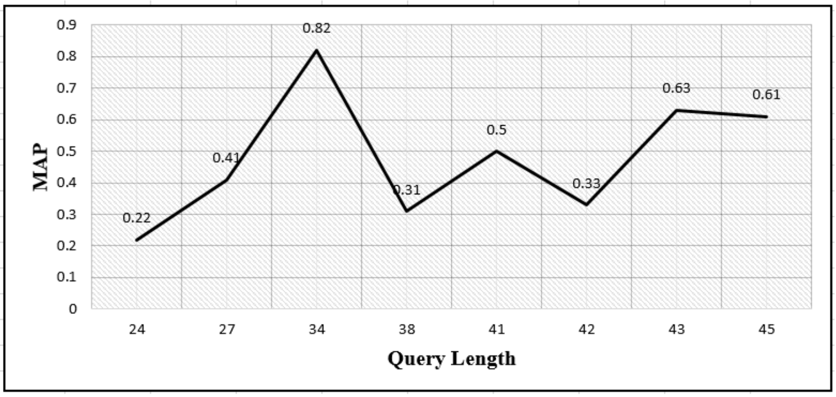


Figure 2.8: Effect of query length on our proposed IR model's retrieval performance for Dataset-1

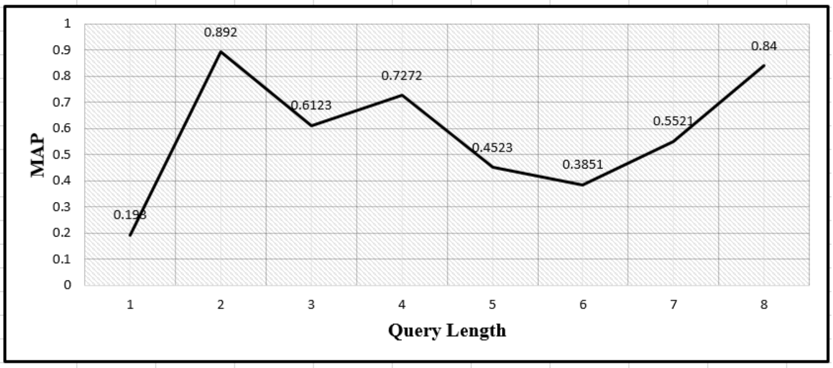


Figure 2.9: Effect of query length on our proposed IR model's retrieval performance for Dataset-2

Error Analysis

We have done an error analysis to inspect where the proposed model may fail. To do this, we have initially identified the queries for which the model performance is dropped. We observe that the model performance for query numbers 106, 117, 118, 122, 124, 83, 86, 87, and 98 from Dataset-1 is degraded. This is because Dataset -1 contains some documents that are not relevant to the query but it contains more query-related words than the relevant documents. In Figure 2.12, we have shown an example query. This is the query number 98, taken from Dataset-1.

Figure 2.12 also shows a document(document number ”1061114_14bdesh3.pc.utf8” in Dataset-1) which is marked as “not relevant” in the ground truth file, but the proposed model ranks the document at the first position in the retrieved list of documents. This is an example of an error made by the proposed model. To inspect the reasons, we manually verify the query-document pair and observe that the document contains many query words

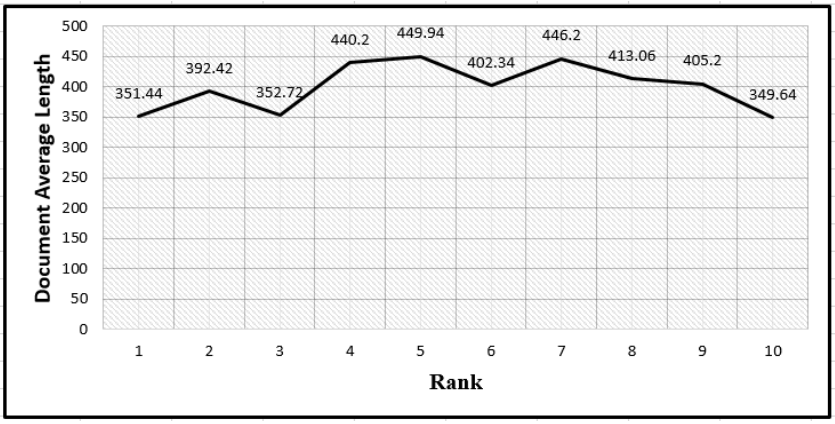


Figure 2.10: Impact of average document length on our proposed IR model's retrieval performance for Dataset-1

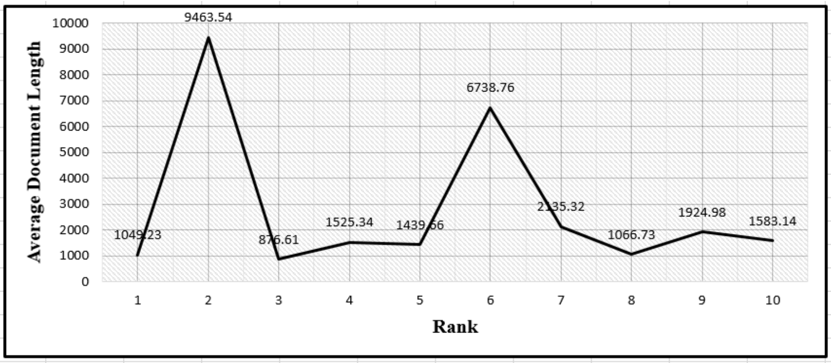
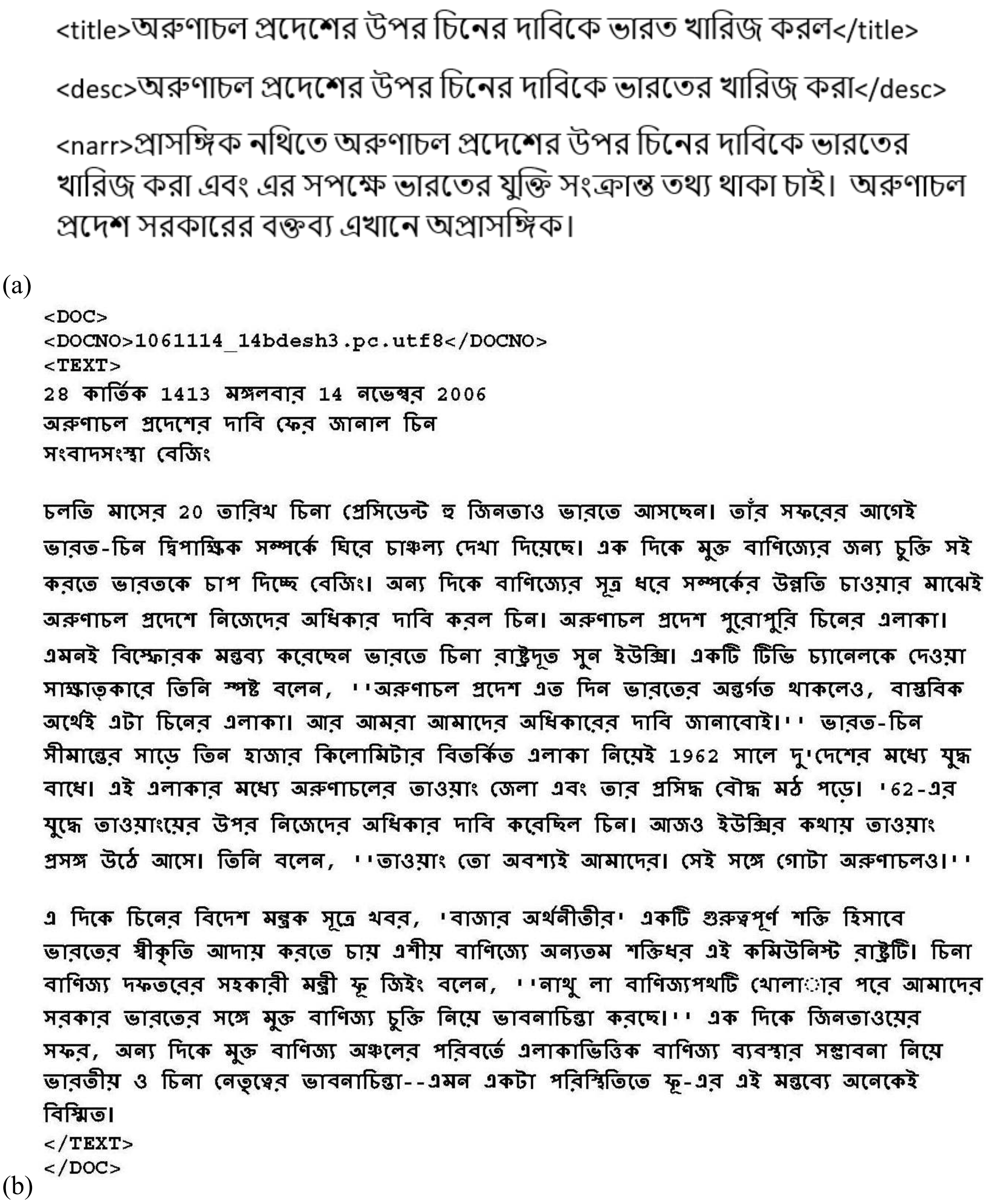


Figure 2.11: Impact of average document length on our proposed IR model's retrieval performance for Dataset-2

like অরুণাচল /Aruṇācala (EN: "Arunachal"), প্রদেশের/Pradēśēra (EN: "of the province"), চিনের/Cinēra (EN: "China's"), ভারত/Bhārata (EN: "India"),
ভারতের/Bhāratēra (EN: "of India"), but the most important query words "kharij", is missing in the document. The proposed smoothing method has also boosted the probability of the query words that are present in the document. As a result, the document has been ranked first in the list though it is not relevant according to the ground truth, Moreover, this document does not contain any argument made by the Indian Government. We think that the more deep semantic analysis is needed to solve such a query-document matching problem because there are many words common between the query and the document, but the document is not relevant to the query according to the ground truth. This is a very complex query situation. To tackle this, the smoothing alone is not sufficient, the model should be able to do conceptual level matching between the query and the document.



<title>অরুণাচল প্রদেশের উপর চিনের দাবিকে ভারত খারিজ করল</title>

<desc>অরুণাচল প্রদেশের উপর চিনের দাবিকে ভারতের খারিজ করা</desc>

<narr>প্রাসঙ্গিক নথিতে অরুণাচল প্রদেশের উপর চিনের দাবিকে ভারতের খারিজ করা এবং এর সপক্ষে ভারতের যুক্তি সংক্রান্ত তথ্য থাকা চাই। অরুণাচল প্রদেশ সরকারের বক্তব্য এখানে অপ্রাসঙ্গিক।

(a)

<DOC>
<DOCNO>1061114_14bdesh3.pc.utf8</DOCNO>
<TEXT>
28 কার্তিক 1413 মঙ্গলবার 14 নভেম্বর 2006
অরুণাচল প্রদেশের দাবি ফের জানাল চিন
সংবাদসংস্থা বেজিং

চলতি মাসের 20 তারিখ চিনা প্রেসিডেন্ট হু জিনতাও ভারতে আসছেন। তাঁর সফরের আগেই ভারত-চিন দ্বিপাক্ষিক সম্পর্কে ঘিরে চাঞ্চল্য দেখা দিয়েছে। এক দিকে মুক্ত বাণিজ্যের জন্য চুক্তি সই করতে ভারতকে চাপ দিচ্ছে বেজিং। অন্য দিকে বাণিজ্যের সূত্র ধরে সম্পর্কের উন্নতি চাওয়ার মাঝেই অরুণাচল প্রদেশে নিজেদের অধিকার দাবি করল চিন। অরুণাচল প্রদেশ পুরোপুরি চিনের এলাকা। এমনই বিস্ফোরক মন্তব্য করেছেন ভারতে চিনা রাষ্ট্রদূত সুন ইউক্সি। একটি টিভি চ্যানেলকে দেওয়া সাক্ষাত্কারে তিনি স্পষ্ট বলেন, ''অরুণাচল প্রদেশ এত দিন ভারতের অন্তর্গত থাকলেও, বাস্তবিক অর্থেই এটা চিনের এলাকা। আর আমরা আমাদের অধিকারের দাবি জানাবোই।'' ভারত-চিন সীমান্তের সাড়ে তিন হাজার কিলোমিটার বিতর্কিত এলাকা নিয়েই 1962 সালে দু'দেশের মধ্যে যুদ্ধ বাধে। এই এলাকার মধ্যে অরুণাচলের তাওয়াং জেলা এবং তার প্রসিদ্ধ বৌদ্ধ মঠ পড়ে। '62-এর যুদ্ধে তাওয়াংয়ের উপর নিজেদের অধিকার দাবি করেছিল চিন। আজও ইউক্সির কথায় তাওয়াং প্রসঙ্গ উঠে আসে। তিনি বলেন, ''তাওয়াং তো অবশ্যই আমাদের। সেই সঙ্গে গোটা অরুণাচলও।''

এ দিকে চিনের বিদেশ মন্ত্রক সূত্রে খবর, 'বাজার অর্থনীতীর' একটি গুরুত্বপূর্ণ শক্তি হিসাবে ভারতের স্বীকৃতি আদায় করতে চায় এশীয় বাণিজ্যে অন্যতম শক্তিধর এই কমিউনিস্ট রাষ্ট্রটি। চিনা বাণিজ্য দফতরের সহকারী মন্ত্রী ফূ জিইং বলেন, ''নাথু লা বাণিজ্যপথটি খোলা◌ার পরে আমাদের সরকার ভারতের সঙ্গে মুক্ত বাণিজ্য চুক্তি নিয়ে ভাবনাচিন্তা করছে।'' এক দিকে জিনতাওয়ের সফর, অন্য দিকে মুক্ত বাণিজ্য অঞ্চলের পরিবর্তে এলাকাভিত্তিক বাণিজ্য ব্যবস্থার সম্ভাবনা নিয়ে ভারতীয় ও চিনা নেতৃত্বের ভাবনাচিন্তা--এমন একটা পরিস্থিতিতে ফূ-এর এই মন্তব্যে অনেকেই বিস্মিত।
</TEXT>
</DOC>

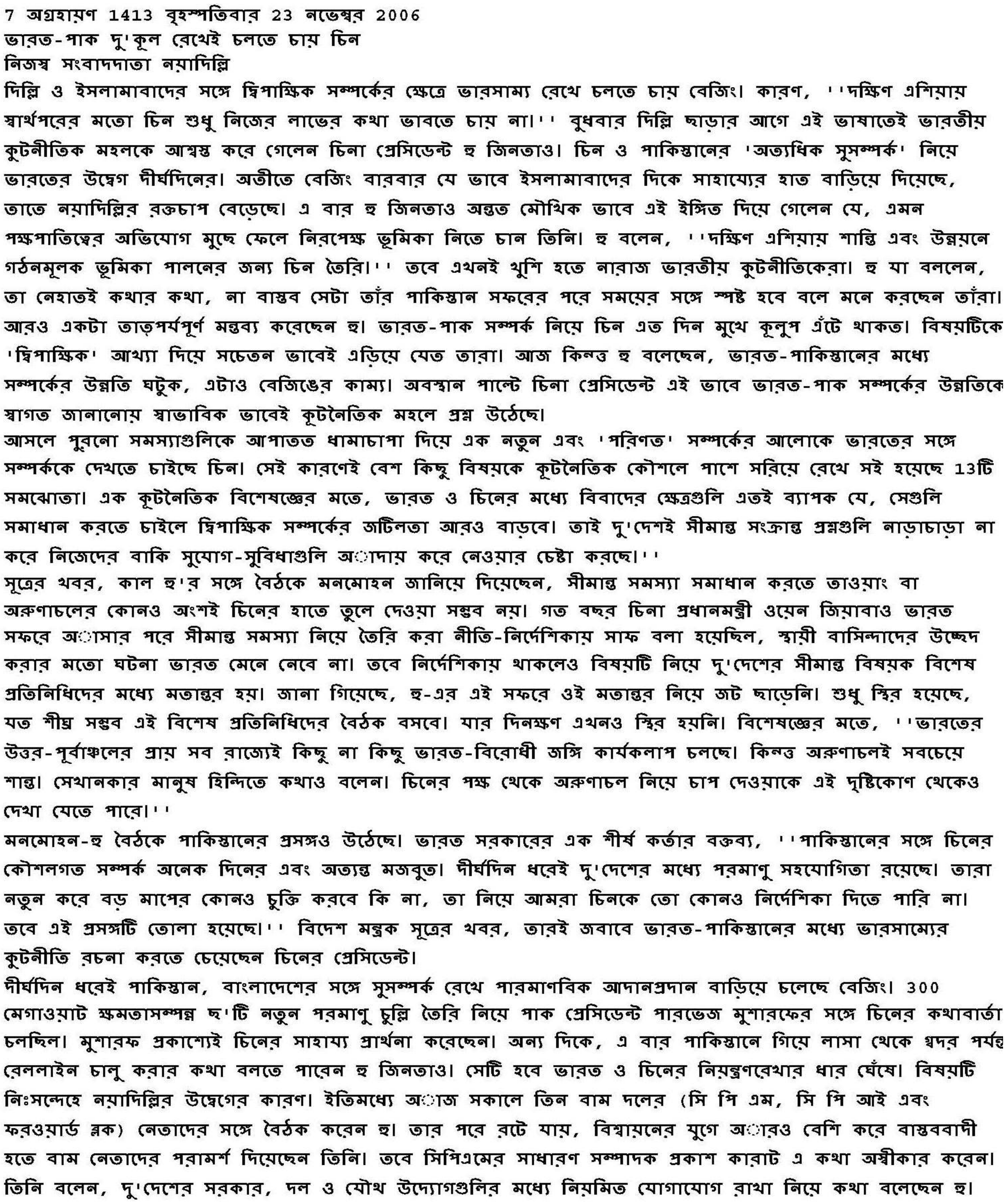
(b)

Figure 2.12: For Dataset-1, query number 98 and an example of a document that is ranked first by the proposed model in represents this query, but it is marked as "not relevant" in the ground truth. Where: (a) Query number: 98 (b) document number: 1061114_14bdesh3.pc.utf8

We also observe another type of error made by the proposed model. In this case, though the document is highly relevant to the query, it appears at the bottom of the ranking list. For example, the document “1061123_23desh2
.pc.utf8” selected from Dataset-1, shown in Figure 2.13 gets a rank of 46 though it is highly relevant to the query number 98 shown in Figure 13. This document contains query words অরুণাচল/Aruṇācala (EN: “Arunachal”), চিনের/Cinēra (EN: “China’s”), ভারত/Bhārata (EN: “India”), ভারতের/Bhāratēra (EN: “of India”), সরকারের/Sarakārēra (EN: ”of the government”), বক্তব্য/Baktabya (EN: ”speech”). The possible reason for such an error is that the document does not contain an important query word প্রদেশ/Pradēśa (EN: ”province”) and the proposed clustering-based smoothing has negatively impacted by reducing the probability estimates for many words to smaller values. Since the document
“1061123_23desh2.pc.utf8” belongs to a relatively larger cluster and the clustering-based language modeling has assigned very small values to the probability estimates for many words.

For Dataset-2, we observe that the model performance for 12 queries is dropped. In Figure 2.14, we have shown document number ”2213” retrieved in response to the query অস্ট্রেলিয়ার বিশ্বকাপজয়ী ব্যাটসম্যান অধিনায়কের অবসর/Asṭreliyāra biśbakāpajayī byāṭasamyāna adhināyakēra abasara (EN:Australia’s World Cup-winning batsman captain retires)[2]. Though this document is marked as “relevant” in the ground truth file for Dataset-2, the proposed model ranks the document at the 469th position in the retrieved list of documents. This is an example of an error made by the proposed model for a test sample chosen from Dataset 2. To investigate the reasons, we manually verified the query-document combination and discovered that the query was too short and the document contained no query words. This is why the proposed smoothing technique was not effective for this short and complex query.

Although the proposed model fails for some queries, we observe that this model performs better than the baseline IR models because the proposed model uses an improved Histogram-based incremental document clustering with a FastText embedding-based document representation. However, the proposed model performs better in many cases. So, the overall performance has improved over the baseline language modeling approach.



7 অগ্রহায়ণ 1413 বৃহস্পতিবার 23 নভেম্বর 2006

ভারত-পাক দু'কূল রেখেই চলতে চায় চিন

নিজস্ব সংবাদদাতা নয়াদিল্লি

দিল্লি ও ইসলামাবাদের সঙ্গে দ্বিপাক্ষিক সম্পর্কের ক্ষেত্রে ভারসাম্য রেখে চলতে চায় বেজিং। কারণ, ''দক্ষিণ এশিয়ায় স্বার্থপরের মতো চিন শুধু নিজের লাভের কথা ভাবতে চায় না।'' বুধবার দিল্লি ছাড়ার আগে এই ভাষাতেই ভারতীয় কুটনীতিক মহলকে আশ্বস্ত করে গেলেন চিনা প্রেসিডেন্ট হু জিনতাও। চিন ও পাকিস্তানের 'অত্যধিক সুসম্পর্ক' নিয়ে ভারতের উদ্বেগ দীর্ঘদিনের। অতীতে বেজিং বারবার যে ভাবে ইসলামাবাদের দিকে সাহায্যের হাত বাড়িয়ে দিয়েছে, তাতে নয়াদিল্লির রক্তচাপ বেড়েছে। এ বার হু জিনতাও অন্তত মৌখিক ভাবে এই ইঙ্গিত দিয়ে গেলেন যে, এমন পক্ষপাতিত্বের অভিযোগ মুছে ফেলে নিরপেক্ষ ভূমিকা নিতে চান তিনি। হু বলেন, ''দক্ষিণ এশিয়ায় শান্তি এবং উন্নয়নে গঠনমূলক ভূমিকা পালনের জন্য চিন তৈরি।'' তবে এখনই খুশি হতে নারাজ ভারতীয় কুটনীতিকেরা। হু যা বললেন, তা নেহাতই কথার কথা, না বাস্তব সেটা তাঁর পাকিস্তান সফরের পরে সময়ের সঙ্গে স্পষ্ট হবে বলে মনে করছেন তাঁরা। আরও একটা তাত্পর্যপূর্ণ মন্তব্য করেছেন হু। ভারত-পাক সম্পর্ক নিয়ে চিন এত দিন মুখে কুলুপ এঁটে থাকত। বিষয়টিকে 'দ্বিপাক্ষিক' আখ্যা দিয়ে সচেতন ভাবেই এড়িয়ে যেত তারা। আজ কিন্তু হু বলেছেন, ভারত-পাকিস্তানের মধ্যে সম্পর্কের উন্নতি ঘটুক, এটাও বেজিঙের কাম্য। অবস্থান পাল্টে চিনা প্রেসিডেন্ট এই ভাবে ভারত-পাক সম্পর্কের উন্নতিকে স্বাগত জানানোয় স্বাভাবিক ভাবেই কূটনৈতিক মহলে প্রশ্ন উঠেছে।

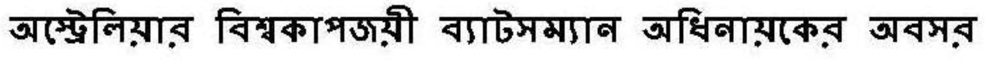
আসলে পুরনো সমস্যাগুলিকে আপাতত ধামাচাপা দিয়ে এক নতুন এবং 'পরিণত' সম্পর্কের আলোকে ভারতের সঙ্গে সম্পর্ককে দেখতে চাইছে চিন। সেই কারণেই বেশ কিছু বিষয়কে কূটনৈতিক কৌশলে পাশে সরিয়ে রেখে সই হয়েছে 13টি সমঝোতা। এক কূটনৈতিক বিশেষজ্ঞের মতে, ভারত ও চিনের মধ্যে বিবাদের ক্ষেত্রগুলি এতই ব্যাপক যে, সেগুলি সমাধান করতে চাইলে দ্বিপাক্ষিক সম্পর্কের জটিলতা আরও বাড়বে। তাই দু'দেশই সীমান্ত সংক্রান্ত প্রশ্নগুলি নাড়াচাড়া না করে নিজেদের বাকি সুযোগ-সুবিধাগুলি অাদায় করে নেওয়ার চেষ্টা করছে।''

সূত্রের খবর, কাল হু'র সঙ্গে বৈঠকে মনমোহন জানিয়ে দিয়েছেন, সীমান্ত সমস্যা সমাধান করতে তাওয়াং বা অরুণাচলের কোনও অংশই চিনের হাতে তুলে দেওয়া সম্ভব নয়। গত বছর চিনা প্রধানমন্ত্রী ওয়েন জিয়াবাও ভারত সফরে অাসার পরে সীমান্ত সমস্যা নিয়ে তৈরি করা নীতি-নির্দেশিকায় সাফ বলা হয়েছিল, স্থায়ী বাসিন্দাদের উচ্ছেদ করার মতো ঘটনা ভারত মেনে নেবে না। তবে নির্দেশিকায় থাকলেও বিষয়টি নিয়ে দু'দেশের সীমান্ত বিষয়ক বিশেষ প্রতিনিধিদের মধ্যে মতান্তর হয়। জানা গিয়েছে, হু-এর এই সফরে ওই মতান্তর নিয়ে জট ছাড়েনি। শুধু স্থির হয়েছে, যত শীঘ্র সম্ভব এই বিশেষ প্রতিনিধিদের বৈঠক বসবে। যার দিনক্ষণ এখনও স্থির হয়নি। বিশেষজ্ঞের মতে, ''ভারতের উত্তর-পূর্বাঞ্চলের প্রায় সব রাজ্যেই কিছু না কিছু ভারত-বিরোধী জঙ্গি কার্যকলাপ চলছে। কিন্তু অরুণাচলই সবচেয়ে শান্ত। সেখানকার মানুষ হিন্দিতে কথাও বলেন। চিনের পক্ষ থেকে অরুণাচল নিয়ে চাপ দেওয়াকে এই দৃষ্টিকোণ থেকেও দেখা যেতে পারে।''

মনমোহন-হু বৈঠকে পাকিস্তানের প্রসঙ্গও উঠেছে। ভারত সরকারের এক শীর্ষ কর্তার বক্তব্য, ''পাকিস্তানের সঙ্গে চিনের কৌশলগত সম্পর্ক অনেক দিনের এবং অত্যন্ত মজবুত। দীর্ঘদিন ধরেই দু'দেশের মধ্যে পরমাণু সহযোগিতা রয়েছে। তারা নতুন করে বড় মাপের কোনও চুক্তি করবে কি না, তা নিয়ে আমরা চিনকে তো কোনও নির্দেশিকা দিতে পারি না। তবে এই প্রসঙ্গটি তোলা হয়েছে।'' বিদেশ মন্ত্রক সূত্রের খবর, তারই জবাবে ভারত-পাকিস্তানের মধ্যে ভারসাম্যের কুটনীতি রচনা করতে চেয়েছেন চিনের প্রেসিডেন্ট।

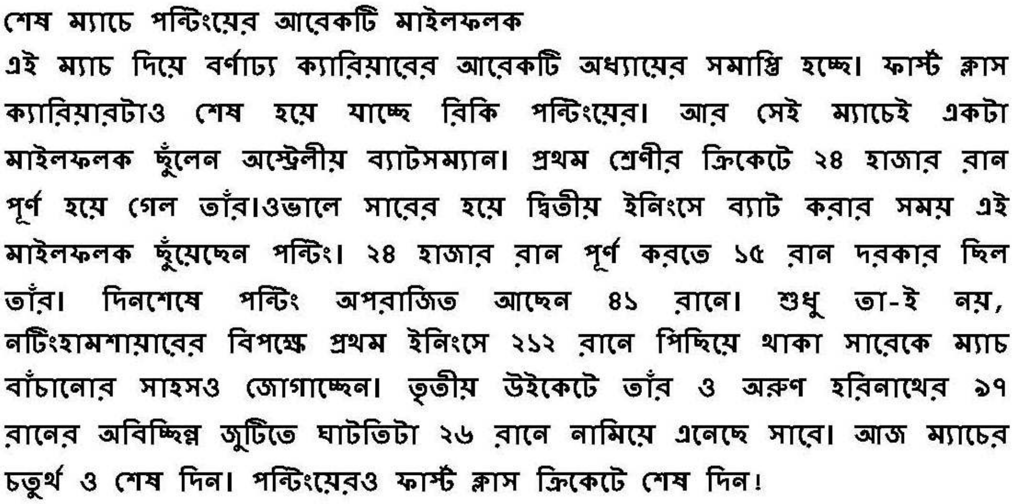
দীর্ঘদিন ধরেই পাকিস্তান, বাংলাদেশের সঙ্গে সুসম্পর্ক রেখে পারমাণবিক আদানপ্রদান বাড়িয়ে চলেছে বেজিং। 300 মেগাওয়াট ক্ষমতাসম্পন্ন ছ'টি নতুন পরমাণু চুল্লি তৈরি নিয়ে পাক প্রেসিডেন্ট পারভেজ মুশারফের সঙ্গে চিনের কথাবার্তা চলছিল। মুশারফ প্রকাশ্যেই চিনের সাহায্য প্রার্থনা করেছেন। অন্য দিকে, এ বার পাকিস্তানে গিয়ে লাসা থেকে গ্বদর পর্যন্ত রেললাইন চালু করার কথা বলতে পারেন হু জিনতাও। সেটি হবে ভারত ও চিনের নিয়ন্ত্রণরেখার ধার ঘেঁষে। বিষয়টি নিঃসন্দেহে নয়াদিল্লির উদ্বেগের কারণ। ইতিমধ্যে অাজ সকালে তিন বাম দলের (সি পি এম, সি পি আই এবং ফরওয়ার্ড ব্লক) নেতাদের সঙ্গে বৈঠক করেন হু। তার পরে রটে যায়, বিশ্বায়নের যুগে অারও বেশি করে বাস্তববাদী হতে বাম নেতাদের পরামর্শ দিয়েছেন তিনি। তবে সিপিএমের সাধারণ সম্পাদক প্রকাশ কারাট এ কথা অস্বীকার করেন। তিনি বলেন, দু'দেশের সরকার, দল ও যৌথ উদ্যোগগুলির মধ্যে নিয়মিত যোগাযোগ রাখা নিয়ে কথা বলেছেন হু।

Figure 2.13: For Dataset-1, an example of a document that is ranked at position 46 by the proposed model in response to this query, but it is marked as “relevant” in the ground truth



অস্ট্রেলিয়ার বিশ্বকাপজয়ী ব্যাটসম্যান অধিনায়কের অবসর

(a)



শেষ ম্যাচে পন্টিংয়ের আরেকটি মাইলফলক

এই ম্যাচ দিয়ে বর্ণাঢ্য ক্যারিয়ারের আরেকটি অধ্যায়ের সমাপ্তি হচ্ছে। ফার্স্ট ক্লাস ক্যারিয়ারটাও শেষ হয়ে যাচ্ছে রিকি পন্টিংয়ের। আর সেই ম্যাচেই একটা মাইলফলক ছুঁলেন অস্ট্রেলীয় ব্যাটসম্যান। প্রথম শ্রেণীর ক্রিকেটে ২৪ হাজার রান পূর্ণ হয়ে গেল তাঁর।ওভালে সারের হয়ে দ্বিতীয় ইনিংসে ব্যাট করার সময় এই মাইলফলক ছুঁয়েছেন পন্টিং। ২৪ হাজার রান পূর্ণ করতে ১৫ রান দরকার ছিল তাঁর। দিনশেষে পন্টিং অপরাজিত আছেন ৪১ রানে। শুধু তা-ই নয়, নটিংহামশায়ারের বিপক্ষে প্রথম ইনিংসে ২১২ রানে পিছিয়ে থাকা সারেকে ম্যাচ বাঁচানোর সাহসও জোগাচ্ছেন। তৃতীয় উইকেটে তাঁর ও অরুণ হরিনাথের ৯৭ রানের অবিচ্ছিন্ন জুটিতে ঘাটতিটা ২৬ রানে নামিয়ে এনেছে সারে। আজ ম্যাচের চতুর্থ ও শেষ দিন। পন্টিংয়েরও ফার্স্ট ক্লাস ক্রিকেটে শেষ দিন!

(b)

Figure 2.14: An example query and a relevant document for Dataset-2 Where: (a) Query number: 98 (b) Document number: 2213.txt

2.5 Chapter Summary

Zero frequency is a fundamental problem in information retrieval using language models and smoothing is applied to deal with this problem. There are various existing smoothing techniques. Out of them, the cluster-based smoothing technique is found to be effective for information retrieval using language models. However, the performance of the clustering-based smoothing technique depends on the cluster quality. In this chapter, we present an improved cluster-based smoothing method that is integrated with a language modeling approach to Bengali information retrieval for improving document retrieval performance. Since the effectiveness of cluster-based smoothing depends on clustering quality, there is scope for improvement by enhancing the clustering algorithm. In this chapter, we present a study on how to improve cluster-based smoothing using a histogram-based incremental clustering algorithm and word embeddings. To our knowledge, this is the first study on the cluster-based smoothing method which is integrated with a language model for developing an effective IR system for the Bengali language which is one of the most spoken Indian languages. The proposed method has been tested on two benchmark Bengali IR datasets. The experimental results show that our proposed model for Bengali document retrieval is effective and it outperforms several baseline IR models, like BM25, the language model, and an existing *K-Mean* clustering-based smoothing model developed by Liu and Croft [101].

However one of the problems of our proposed method is that it does perform well for complex query or abstract queries. We think that the more deep semantic analysis is needed to solve very complex query situations. To tackle this, the smoothing alone is not sufficient, the model should be able to do conceptual-level matching between the query and the document. The semantic approaches can be effective in this situation. In Chapter 3, we discuss the semantic approaches to Bengali IR.

3

Semantic Methods for Bengali Information Retrieval

3.1 Introduction

One of the fundamental problems in information retrieval is the word mismatch problem which arises from the fact that the same question may be asked in different ways using different sets of words. It is also the fact that similar concepts may be presented in different ways in different documents.

We have concluded in Chapter 2 that semantic analysis is needed to solve such a query-document mismatch problem because the traditional IR model cannot detect the relevant documents even though there are many words common between the query and the document. This is a very complex query situation. Smoothing techniques when combined with the document language model can alleviate this problem to some extent. However, this is not alone sufficient. Stemming [7, 6] is also used to improve the recall of IR systems. Moreover, though stemming can alleviate the term mismatch problem by the inflectional word forms, it cannot handle the semantic level word match because many documents that are related to each other semantically but might not share any words appear very dissimilar and occasionally documents that are not related to each other might share common words and appear to be similar. This is due to the nature of the text, where the same concept can be represented by many different words, and words can have different meanings. Therefore, there is a need for a model capable of doing semantic matching between the query and the document.

In the early days, WordNet was used to solve the word mismatch problem [103]. Some researchers used this strategy to improve retrieval performance through query expansion. The synonym, hypernym, and hyponym relations are used to expand the query by adding semantically related terms. But the main drawback of the WordNet-based approach is that the construction of WordNet for a language is a laborious task and WordNet may not contain productive words.

To overcome the limitations of the WordNet-based approach, many researchers have used concept-level matching approaches such as Latent Semantic Indexing (LSI) [28, 34]

[2], word embedding-based approaches [37] [38, 39, 40] [1] that find semantic similarity between words by representing the words in an embedding space [30, 29].

The main drawback of the IR approaches that use simple word vector-based similarity between words is that word vector-based semantic similarity between query words and document words helps to improve recall, but it may hamper precision if word vectors are not created using a huge corpus of documents and they are wrongly positioned in the vector space due to the inaccuracy of the vector creation process. The Word2vec model by the earlier approaches also gives a fixed embedding for a word. This means that a single word has one fixed embedded vector although one word has a different meaning in two different statements due to the context.

To overcome the limitation of the Word2Vec model, recently pre-trained language models [31]. Pretrained language models provide powerful representations of text, reducing the need for extensive hand-crafted features. Pretrained language models, such as Transformer, BERT [31](Bidirectional Encoder Representations from Transformers) have outperformed traditional fixed word embeddings on a variety of NLP tasks [43, 31, 44]. These models can capture rich contextual information, enabling them to understand the meaning of words in the context of a sentence or document. Das et al. [45] used BERT embeddings for document retrieval tasks.

Although many researchers have used LSI, word embedding, and BERT embedding for document representation, limited research has been conducted to apply these methods to Bengali information retrieval and compare these methods using the same benchmark Bengali datasets.

In this chapter, we propose three semantic IR models for the Bengali language and evaluate these models using benchmark datasets. Three methods presented in this chapter are (1) the LSI-based IR model and (2) the word embedding-based IR model and (3) the BERT embedding-based IR model.

In the first part of this chapter, we discuss the query preprocessing method. Then we describe the proposed Latent Semantic Analysis method and Word embedding-based method, the BERT method.

In the second part of this chapter, we describe the benchmark datasets and our own dataset on which the proposed models are tested. The proposed Bengali IR system has been tested on two datasets, The first one is the FIRE dataset which is a well-known benchmark dataset for Bengali information retrieval and the second one is our own dataset developed by Chatterjee and sarkar[1, 33]. Experiments on the benchmark data sets show that our proposed technique achieves state-of-the-art performance for Bengali information retrieval. The evaluation shows that the proposed BERT model can produce improved performance than the other two semantic-based methods.

The rest of the chapter is organized as follows. In section 3.3.3, we present experimental results and parameter tuning. Finally, we conclude in Section 3.4. In this section, we also suggest some future work for further enhancement of the proposed technique.

3.2 Latent Semantic Analysis (LSA) approach for Bengali IR

Latent Semantic Analysis (LSA) is an algebraic-statistical technique for representing meanings of words by their contextual usages and mapping documents into low-dimensional abstract concept space where a concept is represented by the set of words appearing in similar contextual usages. In other words, it maps relations among terms and documents in semantic space. The rationale is that terms that occur in similar contexts will be positioned nearer to each other in the latent semantic space. The degree of relevance between documents and queries is then estimated by computing the cosine measure in the latent semantic space [32, 28]. The above-mentioned IR models suffer more or less from the vocabulary mismatch problem that can be dealt with by the Latent Semantic Indexing (LSI) method [28].

This model starts with a term-document matrix created from the entire corpus of documents, and singular value decomposition (SVD) which is applied to reduce the dimension and construct the latent semantic space, in which the original documents and terms are represented. Then queries that are not part of the original matrix can be folded in by matrix multiplication.

The input to the proposed IR model is a query that is processed first. Query preprocessing involves tokenization, stemming, punctuation removal, and stop-word removal. Stop-words were removed using the list of stop-words provided by FIRE. Finally, the queries are then tokenized into a collection of words. Each query is stemmed using the Bengali stemmer named Yet Another Suffix Stripper (YASS) [6].

Latent Semantic Analysis is done by applying the singular value decomposition (SVD) on a term-by-document matrix B created using the entire corpus of documents. When SVD is applied on B, it produces three matrices that can be combined to produce a low-rank approximation to B.

If r is the rank of the original matrix B and C_k is the corresponding low-rank matrix with the rank k and k is far smaller than r, the dimensions associated with the contextually similar terms are combined (a combination of contextually similar words represents as an abstract concept). As a result, the documents are mapped to a k-dimensional concept space called latent semantic space. When the queries are mapped into the same space, the cosine similarity measure can be used to find conceptual overlap between a document and a query. In this case, if a document and a query share similar concepts will be mapped nearer to each other in the latent semantic space, and the cosine similarity value is considered as the relevance score[28, 96].

The LSI-based document indexing has two important steps: (1) computing a term-by-document matrix, B = $[b_1, b_2, ..., b_d]$, where b_i is the i-th column vector of the term weights for a document and d is the number of documents in the corpus, (2) applying SVD on B.

For the corpus having d documents and the vocabulary size of n, we obtain B whose dimension is nd. Application of SVD on B results in three different matrices M, N and P as Equation (3.1) follows[104]:

$$B_{n\times d} = M_{n\times n}N_{n\times d}P^{T}_{n\times n} \tag{3.1}$$

From the Natural Language processing point of view, in Equation (3.1), M is a term-by-concept matrix. N is a concept-by-concept diagonal matrix and P is a concept-by-document matrix. Three different matrices are shown in Figure. 3.1.

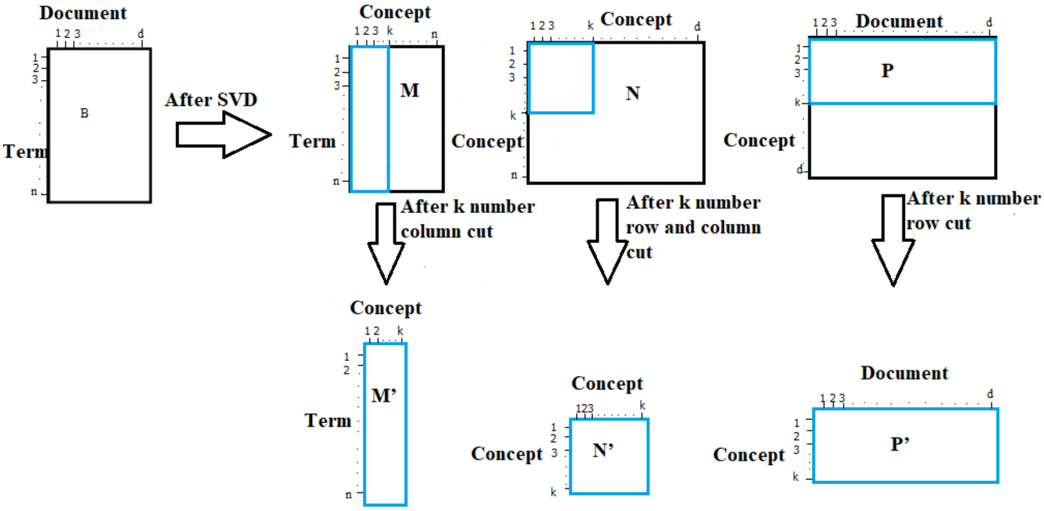


Figure 3.1: SVD of term-by-document matrix

The diagonal of the matrix N contains the singular values measuring the importance of a concept. If the most k-significant singular values are considered, the above-mentioned three matrices are reduced to lower dimensions. This is illustrated in Figure 3.1. A row of the reduced matrix $P' = p_{ij}$, $i = 1tok$ and $j = 1tod$, is indexed by a significant abstract concept, and p_{ij} represents how much the document j is similar with the $i - th$ concept. Transposing P', we obtain $(P')^T$ which is a document-by-concept matrix whose rows are the representations of the documents in the latent semantic space.

For document retrieval, both the queries and documents should be mapped to the same space. For this purpose, a query is initially represented as a vector of the TF-based term weights using the bags-of-words model. Then the projection of this TF-based query vector in the k-dimensional subspace is computed using Equation (3.2).

$$q_k = q_n^T M' N'^{(-1)} \tag{3.2}$$

Where q_n is the query vector. For the $k - th$ query, the relevance score of the $j - th$ document is measured using Equation (3.3) which computes cosine between two vectors.

$$S_j = \frac{q_k \cdot (p'^T)}{\| q_k \| \times \| (p'^T) \|} \tag{3.3}$$

Where $(P'^T)_j$ is the vector representation of the $j-th$ document (the $j-th$ row of (P'^T)) and q_k is the $k-th$ query vector obtained using Equation (3.2).

3.3 Word embedding-based model

Word embedding(WE) is a popular and effective method [29, 36] for word representation and finding the semantic similarity between words. Technically, it represents a word as a vector and it maps the words which are contextually similar nearby in the embedding space. The concept of producing distributed representations is introduced here. They introduce some dependency of one word on the other words intuitively. This dependence would be larger for terms in the context of this word. The word embedding model represents a word as a vector rather than a string of characters. The embedding of the word vectors in k-dimensional space places the word vectors close to each other if they co-occur in similar contexts. Using the word embedding model, based on computing cosine similarities between the word vectors, we can easily identify a list of words that are used in similar contexts with respect to a given word. Word embedding, in contrast to standard word representation, addresses challenges with data sparsity, high-dimensionality, and lexical gaps by capturing semantics and syntactic information in the form of dense vectors. Word embedding has drawn more attention in recent years and has successfully been applied to various NLP applications, including IR.

3.3.1 Word2vec

Word2vec is a natural language processing (NLP) technique that was first published in 2013 [29]. To learn word associations from a large corpus of text, the word2vec algorithm employs a neural network model.

Mikolov et al. [29] focus on distributed representations of words learned by neural networks, it performs significantly better than LSA at preserving linear regularities among words [20, 31]; LSI also becomes computationally very expensive on large data sets.

Semantic Similarity Between Query and a Document

Since we compute semantic similarity between a query and a document based on the semantic similarity between the query words and document words, how to compute semantic similarity between words is an important step. We calculate semantic similarity between a query word and a document word based on cosine similarity between their word vectors. We obtain word vectors for all distinct words in our corpus in the following way:

We have used the Gensim word2vec model under the Python platform. Two important parameters, size of dimension and min count (ignore all words with total frequency lower than this) are set to 50 and 5 respectively. The training algorithm that we have chosen for developing the word2vec model is CBOW (Continuous Bag of words). The other parameters of the model are set to default values.

We locally compute the similarity between a document and a query. For computing similarity between documents and a query, both documents and queries are represented as numeric vectors. We followed the following approach to representing documents and queries as vectors:

Given a query q and a document d, a joint word set T is formed for them. For example, for the following query-document pair, T is the joint word set constructed from the query and the document d. T does not include any redundant words. For example,
Q:" বিরাট কোহলির শীর্ষে সংক্রান্ত তথ্য থাকা চাই"
D:" ক্রিকেটে ব্যাটসম্যানদের রাংকিংয়ে শীর্ষ স্থান দখল করেছেন বিরাট কোহলি"
T= "কোহলির, চাই, তথ্য, বিরাট, শীর্ষে, সংক্রান্ত, কোহলি, ক্রিকেটে, দখল, ব্যাটসম্যানদের, রাংকিংয়ে, শীর্ষ স্থান

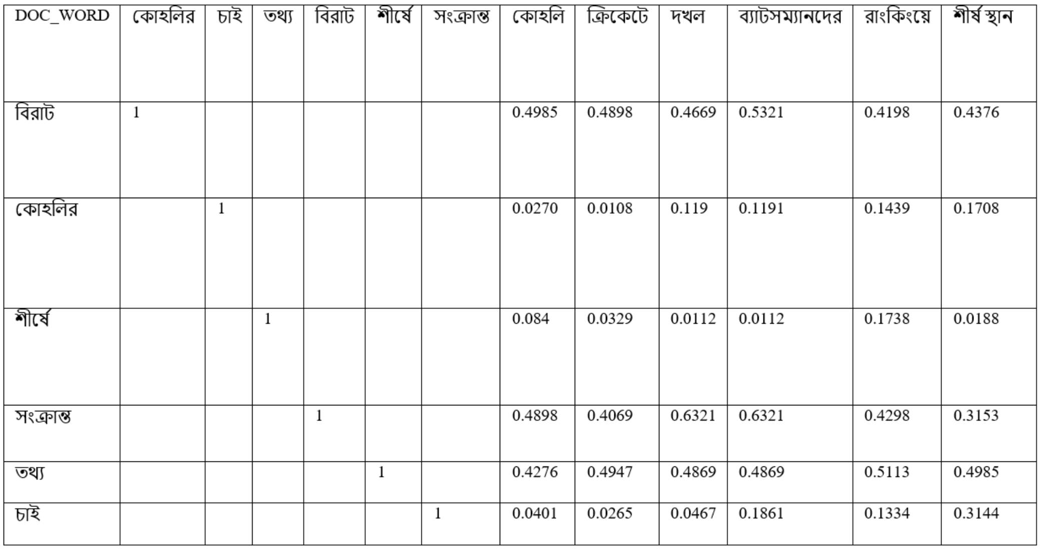
Now the vector derived from the joint word set can be named a lexical semantic vector which has number of entries which is equal to the number of words in the set T. The value of the i th entry ($i = 1, 2, ...n$) into the semantic vector representing the query q or the document d is determined as follows:

- If the stem of the $i - th$ word $t_i$in T exactly matches with at least one of the words contained in the query q, the i^{th} entry is filled with 1.

- Otherwise, compute semantic similarities between the i^{th} word t_i in T and the words contained in the query q. Semantic similarity between a query word and a document word is calculated based on cosine similarity between the word vectors. Say, the maximum obtained similarity value between the word t_i and all query words is T_{max}. If the maximum obtained value, T_{max} >=0.9, the i^{th} entry is filled with the value of T_{max}.

The reason for the introduction of the above-mentioned threshold value (0.9) in our case is to prevent incorporating noise in the semantic vector which is introduced when the maximum similarity scores are very low, indicating that the words are highly dissimilar. Like the method applied to query representation, we also follow the same algorithm stated above to compute the lexical semantic vector for document d.

In Figure 3.3.1, we have shown how the lexical semantic vector is computed. In Figure 3.3.1, the first row shows words in the joint word set T, the first column shows words in the document d. For each word in T, if the same word is also present in d, the cell at the cross point is set to 1 and other entries in that column are kept empty. Otherwise, the cell at the cross point of the most similar word is set to their similarity value or 0, dependent on whether the highest similarity value exceeds the predefined threshold which was set to 0.9 in our experiments.



DOC_WORD	কোহলির	চাই	তথ্য	বিরাট	শীর্ষে	সংক্রান্ত	কোহলি	ক্রিকেটে	দখল	ব্যাটসম্যানদের	রাংকিংয়ে	শীর্ষ স্থান
বিরাট	1						0.4985	0.4898	0.4669	0.5321	0.4198	0.4376
কোহলির		1					0.0270	0.0108	0.119	0.1191	0.1439	0.1708
শীর্ষে			1				0.084	0.0329	0.0112	0.0112	0.1738	0.0188
সংক্রান্ত				1			0.4898	0.4069	0.6321	0.6321	0.4298	0.3153
তথ্য					1		0.4276	0.4947	0.4869	0.4869	0.5113	0.4985
চাই						1	0.0401	0.0265	0.0467	0.1861	0.1334	0.3144

Figure 3.2: llustrate the calculation of lexical semantic vector

For example, the word "ব্যাটসম্যানদের" is not in d, but its similarity with the most similar words কোহালিরis 0.5321. Thus, the cell at the cross point of কোহালির and "ব্যাটসম্যানদের" is set to 0.5321. The lexical vector is obtained by selecting the largest value in each column. As a result, the semantic vector for d is shown in Figure 3.3.1 in the last row. Figure 3.3.1 shows content after applying thresholding over each entry in 3.3.1.

After computing locally, the semantic vectors for query q and document d, we compute the cosine similarity between the query vector q and d as follows :

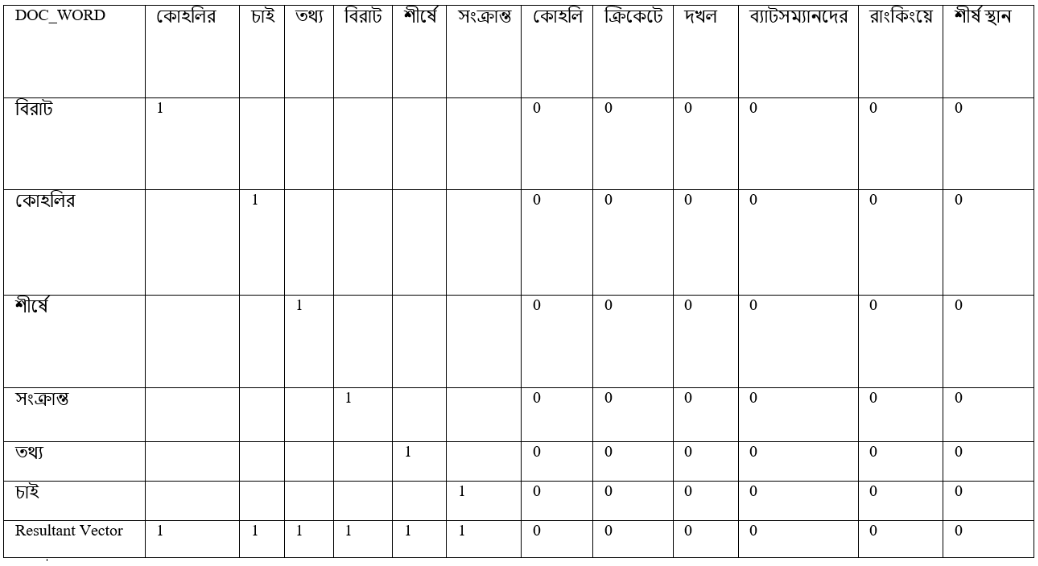
$$Cosine(q, d)\_Score(w, q) = \frac{q \cdot d}{|\, q \,| \cdot |\, d \,|} \tag{3.4}$$

3.3.2 Ranking Documents

For a given query and each document in the corpus, lexical semantic vectors are computed locally for each query-document pair. Then cosine similarity between the corresponding query vector and the document vector is computed.

3.4 BERT-based Semantic Model for IR

Pretrained language models, such as Transformer, BERT [31](Bidirectional Encoder Representations from Transformers) can capture rich contextual information and, enable understanding the meaning of words in the context of a sentence or document. Thus it can gen-



DOC_WORD	কোহলির	চাই	তথ্য	বিরাট	শীর্ষে	সংক্রান্ত	কোহলি	ক্রিকেটে	দখল	ব্যাটসম্যানদের	রাংকিংয়ে	শীর্ষস্থান
বিরাট	1						0	0	0	0	0	0
কোহলির		1					0	0	0	0	0	0
শীর্ষে			1				0	0	0	0	0	0
সংক্রান্ত				1			0	0	0	0	0	0
তথ্য					1		0	0	0	0	0	0
চাই						1	0	0	0	0	0	0
Resultant Vector	1	1	1	1	1	1	0	0	0	0	0	0

Figure 3.3: llustrate the calculation of lexical semantic resultant vector

erate powerful representations of queries and documents. The BERT model is pre-trained on large-scale text corpora and learns linguistic patterns in language. BERT is developed to pre-train deep bidirectional representations from the unlabeled text by conditioning on both left and right context simultaneously across all layers. The previous research works [43, 31, 44] showed that this model is more effective than standard word embeddings in the semantic representation of texts.

The proposed BERT-based IR model follows a two-step process. In the first step, an initial search is performed using the BM25 IR model[18]. In the second step, BERT is used for query and document representation, and the documents retrieved by BM25 are re-ranked based on the similarity scores computed for query-document pairs. At the time of the Initial search using the BM25 IR model, we select the top k (which is set to 100) documents from the initial search results for re-ranking. Re-ranking is done based on a hybrid score which is a combination of BM25 score and BERT score. An architecture of the BERT-based Semantic Model for IR is shown in Figure 3.4.

For the initial search, We have used the BM25 score function which is used by the Lemur Project [8] is shown in Equation (1.1). This score function assigns a relevance score to document d with respect to the query q. The details of the BM25 model are already described in Chapter 1. This model has several tuning parameters which are $K1$, $K3$, and B. For our settings, we set the values of these parameters as $K1$=2.2, B=0.3, and $K3$=250.

For the BERT model, we have made use of Bangla-Bert-Base [105], which uses mask language modeling for Bengali information retrieval. Bangla-Bert was trained using code from the GitHub repository of Google BERT.[1]. The model architecture of the bert-base-

[1] https://github.com/google-research/bert

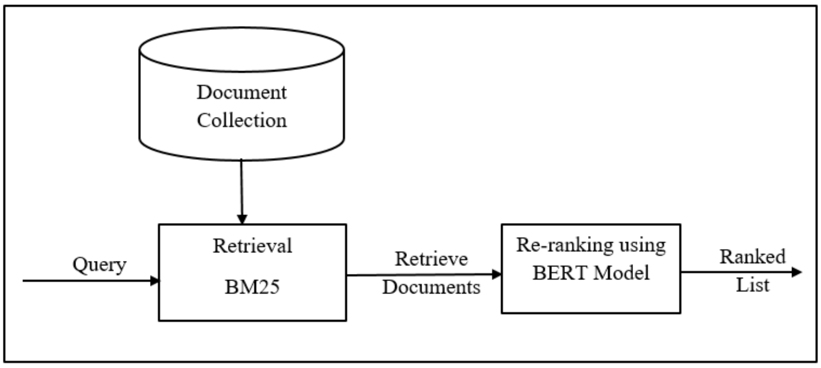


Figure 3.4: BERT-based Semantic Model

uncased contains 12 layers, 768 hidden units, 12 heads, 110M parameters.

Since the Bangla-Bert-Base model was not designed for processing long documents[106] and it can process the first 512 tokens as input for document representation, we apply the BIRCH algorithm [106] to find the better document representation. In the BIRCH algorithm, sentence-level information is aggregated for document representation. First We represent each sentence as a vector using the Bangla-Bert-Base. The query is also represented In the same way. Then we calculate the cosine score between the sentence vectors and the query vector, and the top 3 sentences are selected based on the similarity scores. The average of the cosine scores for the top 3 sentences is considered a BERT score for a document-query pair. However, only using the BERT score for reranking does not give better results. To achieve better results we combine the BERT score with the BM25 score using Equation (3.5) to produce the hybrid score which is used for re-ranking.

$$Score(d) = a \cdot BM25\_Score(d) + (1 - a) \cdot BERT - based\_score(d) \tag{3.5}$$

Where $Score(d)$ is the hybrid score, $BM25\_Score(d)$ is the relevance score assigned by the BM25 model and $BERT - based\_score(d)$ is the relevance score assigned by the BERT-based model.

3.4.1 Re-Ranking Documents

For a given query and each document in the corpus, the combined score is computed using Equation (3.5), and the documents are ranked based on their hybrid scores.

3.5 Evaluation, Experiments, and Results

The three semantic models proposed in this chapter are tested on the following two Bengali datasets.

- Dataset used in [1], contains 19 queries to search relevant documents from a corpus of approximately 3255 documents. These 3255 documents are collected from Anandabazar Patrika's online pages. This corpus contains political news, sports news, International news, district-wise news of West Bengal, and Entertainment news.

- FIRE Dataset (which was mentioned as Bengali Dataset-1 in Chapter 2) contains 123021 documents and a relevance file. The relevance file contains 50 queries(query numbers 76 to 125) and the human relevance judgment for each query.

3.5.1 Evaluation

To evaluate the retrieval model's effectiveness, we have used a well-known technique, mean average precision(MAP) which is already described in the chapter (subsection 1.6.1).

3.5.2 Experiments

We have conducted the following experiments to prove the effectiveness of the proposed semantic Bengali IR models

Experiment-1:
In the first experiment, we implemented the latent semantic indexing(LSI) based IR model for Bengali information retrieval that uses semantic representation for document and query representation using the method described in section 3.2.

Experiment-2:
In the second experiment, we used the word embedding-based IR model. In this case, fixed word embedding obtained using Word2Vec model is used to represent query and document. The details of this model are described in section 3.3.

Experiment-3:
In the third experiment, we used the BERT embedding-based re-ranking model. The initial search is performed first using the BM25 IR model. Then we select the top 100 documents from the search results returned by BM25. The search results selected in the first stage are re-ranked by the BERT-based model to improve the MAP score for the IR model. In this case, documents are re-ranked using the combined scores where the BERT score is combined with the BM25 score to obtain the combined score. The BERT score is calculated as the cosine similarity between the query vector and the document vector obtained from the BERT model. The Details of document representation and query representation using the BERT model are described in section 3.3.

Experiment-4:

The fourth experiment is almost similar to the third experiment with the exception that, In this case, documents are re-ranked using BERT scores only. The reason for conducting this experiment is to prove the effectiveness of only the BERT score in improving the re-ranking process.

3.5.3 Results and Discussion

Table 3.1 displays the MAP scores obtained by our developed semantic IR models on the Dataset used in [1]. The results shown in Table 3.1 reveal that, out of the semantic models proposed in this chapter, the BERT-based model with RE-ranking using combined scores (BM25 score + BERT-score) performs better than the word embedding-based model and the LSI-based IR model. Surprisingly, we observe that the re-ranking using only the BERT score does not perform better than the LSI model although it performs better than the fixed word embedding-based model.

Table 3.1: Performance Comparisons of the proposed IR models on the Dataset used in [1]

Model Name	MAP Score
LSI-based IR Model	0.5078
Word embedding-based IR Model	0.45
BERT-based model with re-ranking using only BERT score	0.4828
BERT-based model with RE-ranking using BM25 score + BERT-score	0.5869

Table 3.2 shows the MAP scores obtained by our developed semantic IR models for the FIRE Dataset which is the benchmark dataset for Bengali IR. As we can see from Table 3.2 the BERT-based model with RE-ranking using combined scores (BM25 score + BERT score) performs better than the word embedding-based model and the LSI-based IR model. By analyzing the results obtained on both datasets, we can observe that the BERT-based model with RE-ranking using combined scores (BM25 score + BERT score) performs the best.

Table 3.2: Performance Comparisons of the proposed semantic IR models on the FIRE Dataset

Model Name	MAP Score
LSI-based IR Model	0.4325
Word embedding-based IR Model	0.3023
BERT-based model with re-ranking using only BERT score	0.3940
BERT-based model with RE-ranking using BM25 score + BERT-score	0.4906

Parameter tuning

In this subsection, we present how we obtain the optimal values for the tunable parameters of the proposed models.

Parameter tuning for the LSI Model
We observed that the most important parameter whose values affect the retrieval performance of LSI based IR model is k. K indicates the dimension of semantic space into which documents and queries are mapped, we have tuned this parameter to achieve better results. The effect of varying k value on the MAP score for LSI based model is shown in Figure 3.3. It indicates that the LSI-based model with k set to 95 gives the best MAP score.

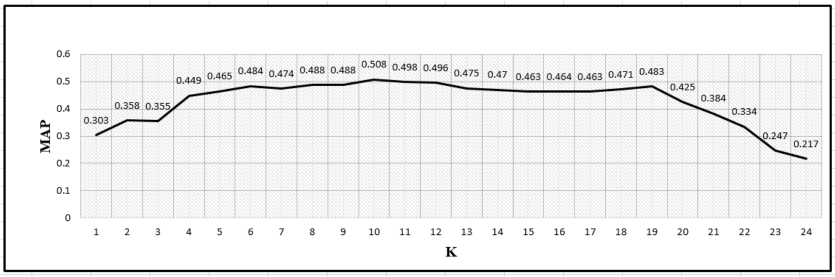


Figure 3.5: Impact on MAP scores for LSI-based model when k values are varied on the Dataset [1]

For the FIRE Dataset, the model parameter k is set as 500, for which we obtain the best results

Parameter tuning for the word2vec Model
One of the important parameters in the Word embedding-based model is the word similarity threshold which determines whether a query word and a document word are semantically similar or not. We have shown the effect of this threshold value in Figure 3.4. It is also evident from Figure 3.4 that the word embedding-based model gives the best results when the word similarity threshold is set to 0.9.

For the FIRE Dataset, the model parameter *m* is set as 0.9, and the word similarity threshold is set to the values for which we obtain the best results on Dataset[1].

Parameter tuning for the BERT-based model with RE-ranking using BM25 score + BERT-score Model
One of the important parameters in the BERT-based model with RE-ranking using the BM25 score + BERT-score Model is the blending parameter. We observed that the most important parameter whose values affect the retrieval performance of the BERT-based model with RE-ranking using the BM25 score + BERT-score Model is a. We have shown in Figure 3.5 the effect of the blending parameter (weight) on the performance of our proposed combined model. It is evident from Figure 3.5 that we get the best results on our data set

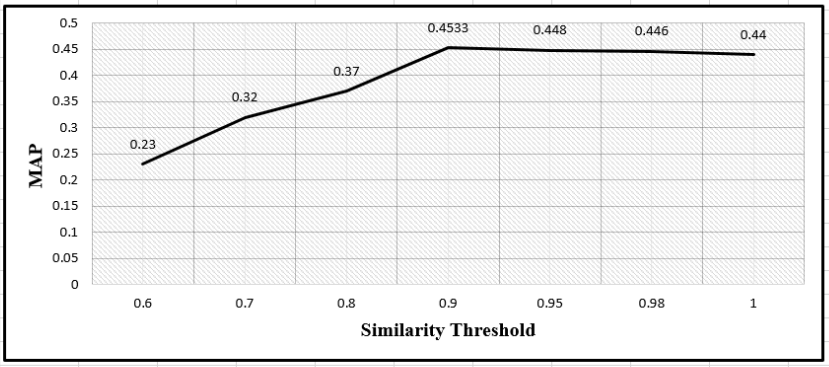


Figure 3.6: Impact on our proposed model's retrieval performance on Dataset [1] when m is varied

when the value of the blending parameter a is set to 0.8. which determines whether.

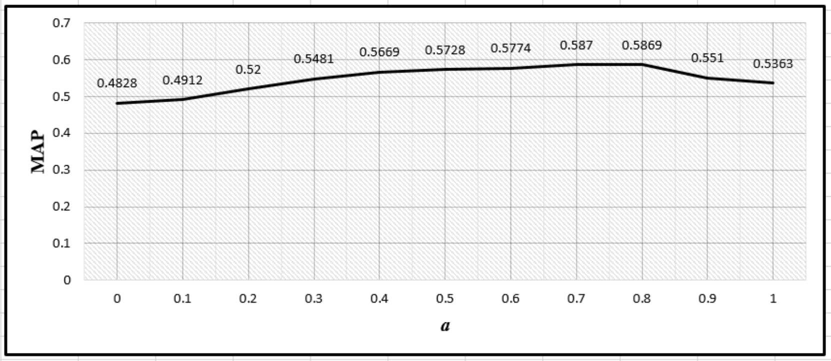


Figure 3.7: Impact on MAP scores for RE-ranking using the BM25 score + BERT-score Model when a values are varied on the Dataset [1]

Comparison with existing methods

We have chosen some existing models for comparison with our proposed best model. We have compared our proposed best model with the BERT-based IR model proposed in [45], which is closest to our work. The main difference between our proposed model and the model proposed in [45] is that we have used an effective BM25 model instead of using TFIDF-based vector space model(VSM) for retrieving the initial search results which are further re-ranked using the BERT model. Das et al. [45] used the TFIDF-based VSM for producing initial search results whereas we have used a BM25 model for producing the initial search results. Since BM25 is a more effective retrieval model than the TF-IDF VSM model, this results in a better retrieval performance when re-ranking is done using the BERT-based process. Moreover, Das et al.[45] tested their model on a smaller dataset.

A performance comparison of our proposed best model(BERT-based model with RE-ranking using BM25 score + BERT-score) with the other baseline models is shown in Table 3.3. The results show that our proposed IR model performs significantly better than the baseline models it is compared to.

We have calculated the percentage improvement in the performance of our proposed approach over the baseline models using Equation (2.20) and we present percentage improvement results in Table 3.3.

Table 3.3: Percentage improvement achieved by our proposed best model over the baseline model using FIRE Dataset

Model Name	Dataset	% Improvement
`BERT-based Re-ranking IR Model[45]`	FIRE Dataset	8.25
`BERT-based Re-ranking IR Model[45]`	Dataset[1]	7.2

Table 3.3 shows the best performance improvement achieved by our proposed best IR model over the baseline model[45]. Performance improvement over the baseline model for the FIRE Dataset is 8.25% and for the Dataset[1] 7.2%.

3.6 Chapter Summary

In this chapter, we present mainly three types of semantic IR models: the LSI-based semantic IR model, the fixed word embedding-based semantic IR model, and the BERT-based semantic IR model to overcome the word mismatch problem. The experimental results suggest that the IR model that uses BM25 for generating initial search results and re-ranks them based on a combination of BM25 and BERT scores performs the best among all semantic IR models presented in this chapter. The best semantic BERT-based reranking IR model should be a two-step ranking-based IR model where, in the first step, documents are ranked using the traditional IR model, BM25, and in the second step, the documents returned at the first step are re-ranked using BM25 score +BERT-score. The reason for this combination is that the BERT score itself is not enough to re-rank the documents because it does not capture the relative importance of query words present in the documents. The proposed model also outperforms a recently published IR model that has used BERT-based semantic representation.

Our next plan is to explore a query expansion approach for Bengali information retrieval.

4 Hybrid Semantic Query Expansion for Bengali Information Retrieval

4.1 Introduction

In the early days, two influential IR models adopted by the researchers for the Bengali IR were the vector space model (VSM) [10] and the statistical language model[20]. The main drawback of this type of IR model is that it ignores the relevant documents if there is no query term match between the documents and the given query. This problem is known as the word mismatch problem or vocabulary mismatch problem, which occurs when the query is relatively short or the same topic is posed in various ways. Furthermore, it is a fact that related ideas can be presented in many ways in various texts. This leads to the vocabulary mismatch or the term mismatch problem. As a result, IR systems are unable to find documents that match the user's needs. Query expansion (QE) [4] is a well-known effective approach to address this problem. Although earlier research [7, 26, 8, 6] on Bengali IR incorporated various term normalization methods like stemming or lemmatization into the traditional vector space model to deal with this problem, such methods fail to detect semantic match.

Query expansion (QE) is a process that selects and adds additional terms to a query to alleviate the word mismatch problem, and hence improve the retrieval performance[46]. In recent times, word embedding (WEs) [32] has impacted immensely almost all-natural language processing(NLP) tasks and it can detect the semantic match between words. WE is a process of representing vocabulary terms as real-valued dense vectors. WE is useful in computing the semantic relatedness between words. Some research works have already been carried out to show the effectiveness of word embeddings in query expansion[107, 108]. But some recent works on query expansion have shown that instead of using a single query expansion technique, the fusion of multiple expansion techniques is useful in extracting more informative query expansion terms [62, 63, 64, 65, 66].

In this chapter, we propose a novel hybrid query expansion framework that combines statistical, lexical, and word embedding-based semantic methods to choose the contextually and semantically related terms for query expansion leading to improving Bengali retrieval performance. The proposed approach has multiple stages. In the first stage, the initial search is performed using okapi BM25 [19]. In the second stage, the candidate expansion terms are extracted using three methods (1)a lexical method that uses a Bengali WordNet, (2) a word embedding-based method that uses the top $k_d^{(}(emb))$ documents from the initial search results and applies the word-to-word semantic similarity measure to extract the words which are highly similar to the query words, and (3) a term frequency-based method that uses term frequency statistics for extracting words from the initial search results. At the third stage of the proposed approach, the candidate expansion terms extracted using three different extraction methods are combined to create a pool of candidate expansion terms which are ranked based on their scores where the score of a candidate term is a linear combination of its contextual score and frequency score. After ranking the candidate terms, a certain number of top-ranked terms are selected as the final expansion terms and they are added to the original query, and the search is again performed using the expanded query.

In this chapter, we introduce a novel hybrid query expansion approach for Bengali information retrieval and evaluate these models using benchmark datasets. Three models are developed by considering all possible combinations of the three expansion term extraction methods, (1) expansion term extraction using WordNet,(2) expansion term extraction using word embedding, and (3) expansion term extraction using term frequency and 4 hybrid models presented in this chapter.

In the first part of this chapter, we discuss the query pre-processing and document preprocessing methods. In the second part of this chapter, we describe the proposed Okapi BM25 IR models. In the third part of this chapter, we also describe the Candidate term extraction methods for query expansion. In this part, we also describe how candidate terms extracted by the individual term extraction method are combined to create a pool of candidate terms. The candidate terms in the pool are then ranked based on the contextual score and the frequency score for selecting the final expansion terms. In the third part of this chapter, we describe the benchmark datasets on which the proposed model is tested. The rest of the chapter is organized as follows. In section 4.4, we evaluate our proposed models using a benchmark dataset. Experiments on the benchmark data set show that our proposed technique achieves state-of-the-art performance for Bengali information retrieval. Finally, we conclude in Section 4.5. In this section, we also suggest some future work for further enhancement of the proposed technique.

4.2 Methodology

To improve the performance of the Bengali information retrieval system, we have proposed a hybrid query expansion method that combines the lexical, semantic, and statistical methods. The lexical method involves WordNet in extracting the candidate expansion terms synonymous with the query terms whereas the semantic and the statistical methods are incorporated into the pseudo-relevance feedback framework for extracting the candidate expansion terms. The statistical method uses term frequency statistics to extract candidate terms from some highest ranked k documents selected from the initial search results whereas the semantic method extracts terms based on word vector similarity between the original query terms and the terms appearing in those k documents. We have applied the well-known Okapi BM25 IR model for producing the initial search results and used the top-rated documents retrieved by this model as the input to the proposed hybrid query expansion method. A Bengali WordNet[1] is connected to the query expansion module. Thus our proposed hybrid query expansion model generates three sets of candidate expansion terms, (1) synonym set, (2) semantic set, and (3) frequent set. Using these three sets of candidate expansion terms, a pool of candidate terms is created. Then each candidate expansion term is assigned a score which is obtained by linearly combining the context score and frequency score. The contextual score is computed using the term's similarity with the query vector which is the sum of word embedding vectors corresponding to the query words. The frequency score, on the other hand, is determined by counting the instances of a term's stem in the top-ranked documents selected from the initial search results. The top-scoring candidate expansion terms are added to the initial query after the candidate expansion terms are ordered according to their scores. Finally, the search is again performed by submitting the expanded query to the Okapi BM25 IR model.

Our development process of the IR model with the proposed query expansion method is composed of seven main steps.

Step 1. Query and Document Pre-processing

Step 2. Implementing Okapi BM25 IR model

Step 3. Candidate term extraction for query expansion

Step 3.1 Wordnet-based candidate expansion term extraction

Step 3.2 Word embedding-based candidate expansion term extraction

Step 3.3 Frequency-based candidate expansion term extraction

Step 4. Combining all types of candidate terms to form a pool of the candidate terms.

Step 5. Rank the candidate terms according to their scores where a score of the term is a linear combination of the contextual score and the frequency score.

Step 6. Select the higher-scoring terms as final terms and add them to the first query to form an expanded query.

Step 7. Performing search again by submitting the expanded query to the Okapi BM25 IR model.

The architecture of our developed IR system is shown in Figure 4.2. We discuss separately each step of the proposed IR model in this section.

[1] `https://pypi.org/project/banglanltk/`

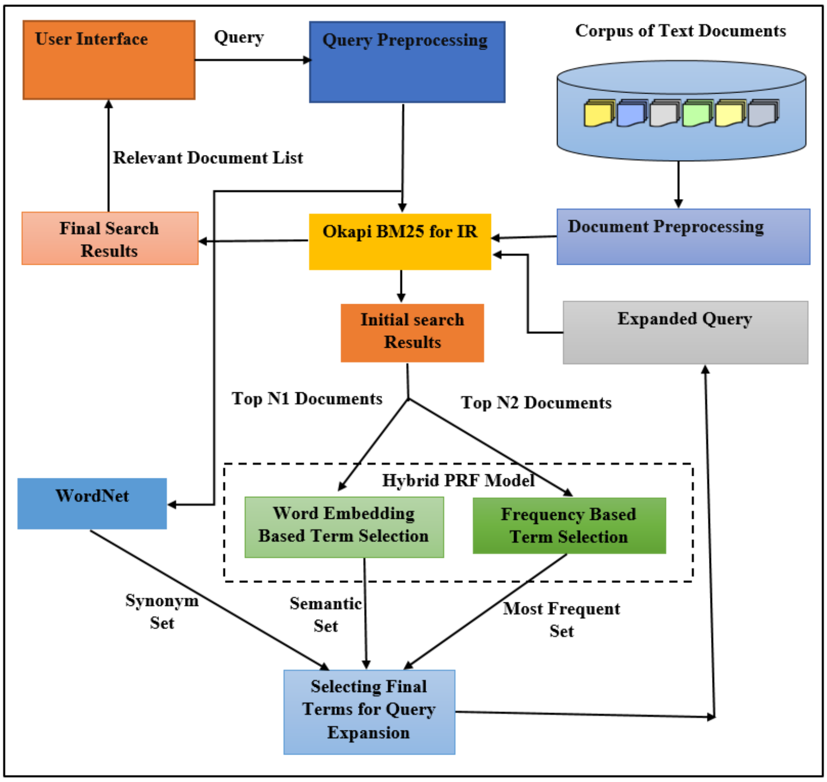


Figure 4.1: Proposed Model Architecture

4.2.1 Query and Document Preprocessing

The proposed IR model's input consists of a query and a set of documents. Query and document preprocessing involve tokenization, stemming, punctuation removal, and stop-word removal. For stop-word removal, we have used the list of stop-words provided by FIRE. After punctuation and stop-word removal, each word is stemmed using the Bengali stemmer named Yet Another Suffix Stripper (YASS) [6].

4.2.2 Okapi BM25 IR Model

The Okapi BM25 IR model, also known as BM25[18] was employed as the basic IR model for retrieving the relevant documents from the collection of documents. The search is performed by the Okapi BM25 IR model using the original query and the expanded query. The Lemur Project's [8] score function in BM25 is used to rate the relevance of document d in relation to the query q. The equation for the score function is as follows.

$$BM25\_SCORE(d,q) = \sum_{q_i \in q} \frac{K3+1}{K3} \times IDF(q_i) \times TF\_factor(q_i,d) \quad (4.1)$$

where

$$TF\_factor((q_i,d)) = \frac{TF(q_i,d) \times (1+K1)}{TF(q_i,d) + K1 \times ((1-B) + B \times \frac{Doc\_Len(d)}{Avg\_Doc\_Len}))} \quad (4.2)$$

Where:
$TF(q_i,d)$) is the frequency of the word q_i in the document d.
$Doc\_len(d)$ is the length of document d.
Avg_Doc_Len is the average document length in the corpus. The Inverse Document Frequency (IDF) of the word q_i is calculated in terms of DF as:

$$IDF(q_i) = log[0.5 + \frac{N}{DF(q_i)}] \quad (4.3)$$

Where $DF(q_i)$ is the document frequency that counts the number of documents in the collection C that contain the given word q_i.
N represents the total number of documents in collection C.
$K1$, $K3$, and B are the tuning parameters. For our experiment, we set the values of these parameters as $K1$=2.2, B=0.3, and $K3$=250.

Initial search with the original query (without expansion) is performed first using the BM25 IR model, and then we select the top k documents from the search results returned by BM25 for utilizing them in the candidate query term extraction process for query expansion. The value of k is different for the different expansion methods and its optimal value for our task is experimentally determined. In this chapter, the count of the highest ranked relevant documents used by the frequency-based expansion method is indicated by $\mathrm{k}_{\mathrm{d}}^{(freq)}$ and for the word embedding-based expansion method, it is indicated by $\mathrm{k}_{\mathrm{d}}^{(emb)}$. Since both methods extract candidate expansion terms from the top-ranked documents chosen from the first search results, they are actually two different variations of the PRF(pseudo-relevance feedback) approach. We have also incorporated WordNet for extracting terms that are synonymous with the original query terms and added them to the pool of the candidate query terms.

4.2.3 Candidate expansion term extraction

As we mentioned earlier, we have used three methods for extracting the candidate terms for query expansion:- (1) WordNet-based candidate expansion term extraction, (2) Word embedding-based candidate expansion term extraction, and (3) Frequency-based candidate expansion term extraction. These three methods are discussed in this subsection.

Wordnet-based candidate expansion term extraction

After pre-processing the original query, we get the list of query words. Then we take each word from a query word list and find their synonyms from WordNet available in the bangla nltk toolkit[2] under the python platform. These synonyms are added to the pool of candidate expansion terms. In Figure 2, we have referred to it by the name ”synonym-set”.

Word embedding-based candidate expansion term extraction

We employ a semantic method that considers the contextual similarity between the original query words and the words in the top $\mathrm{k}_{\mathrm{d}}^{(emb)}$ relevant documents returned by the initial search process. We used a pre-trained word embedding model to find the vectors for each word. The word embedding model represents each word as a k-dimensional vector in an embedding space. However, for calculating semantic similarity between any two words, we use a cosine similarity measure that uses word vectors given by the word embedding model and gives a similarity score.

To obtain the vector for each word, we used the pre-trained model created by [41] for the Bengali language. Data from Wikipedia and Common Crawl were used to train this pre-trained model. Each word vector has a dimension of 300. We select the top $\mathrm{k}_{\mathrm{d}}^{(emb)}$ documents from the initial search results to extract from them the terms that have semantic relations with the initial query words. The terms extracted by this process are added to the pool of candidate terms. In Figure 2, we have referred to it by the name ”semantic-set”. The cosine similarity measure is used to find the similarity between a query word and a word in the collection of top $\mathrm{k}_{\mathrm{d}}^{(emb)}$ documents. The words whose similarities with the query word pass a predefined threshold value(threshold value of 0.7) are added to the pool of candidate query terms. The value of $\mathrm{k}_{\mathrm{d}}^{(emb)}$ is experimentally determined and $\mathrm{k}_{\mathrm{d}}^{(emb)}$ is set to 10 for the best results.

Frequency-based candidate expansion term extraction

Many existing pseudo-relevance feedback approaches to query expansion have used frequency statistics of the words present in the feedback documents for extracting the important terms as the candidate expansion terms[109]. We have also chosen the most frequent terms from top-rated documents retrieved by the initial query as the candidate expansion terms and they are added to the pool. We have chosen the most frequent m_{f} distinct terms from the top $\mathrm{k}_{\mathrm{d}}^{(freq)}$ documents. For the best results, the values of m_{f} and $\mathrm{k}_{\mathrm{d}}^{(freq)}$ are set to

[2] https://pypi.org/project/banglanltk/

30 and 5 respectively. While calculating the word (term) frequency, we consider the frequency of the word's stem as the frequency of the word. We have used YASS stemmer[6] for stemming the words.

4.2.4 Ranking Candidate Expansion Terms and Selecting Final Set of Expansion Terms

We create a pool of candidate expansion terms using the three automatic methods discussed above in this section. Thus the pool of candidate expansion terms is as follows.

Candidate Expansion Terms Pool = Synonym-set ∪ Semantic-set ∪ Most frequent-set

An automatic process for expansion term extraction may give rise to a new problem called query drift[4]. For example, if the query is related to gold mines and the top few documents are related to gold mines in Indonesia, then this may lead to query drift directing the search toward the documents on Indonesia. To avoid this situation, we need to carefully choose the final set of expansion terms from the candidate pool. For this purpose, the terms in the candidate pool are compared with the contextual representation of the entire source query. We define a ranking function for selecting expansion terms from the candidate pool. According to this function, the term that is more frequent in the top-ranked feedback documents, as well as more similar to the entire context of the original query, is assigned a higher score indicating its worthiness of being an expansion term. For example, According to our defined ranking function given in Equation (4.6), the synonyms will be chosen as the expansion terms if they are highly frequent in the feedback documents and contextually similar to the initial query.

For the contextual representation of a query, the word vectors of the pre-processed query are averaged as follows.

$$V(q) = \sum_{q_i \in q} WordToVec(q_i) \tag{4.4}$$

where $V(q)$ is the contextual vector for the query q.

The contextual score of a candidate term w is computed using Equation (4.5) which gives a value of the cosine similarity between the word vector of w and $V(q)$.

$$contextual\_Score(w, q) = \frac{WordToVec(W) \cdot V(q)}{\mid WordToVec(W) \mid \cdot \mid V(q) \mid} \tag{4.5}$$

The final ranking function for the candidate expansion terms is defined in Equation

(4.6) by combining the contextual score and frequency score.

$$Score(w) = \alpha \times contextual\_score(w, q) + (1 - \alpha) \times frequency\_score(w) \quad (4.6)$$

Where: $Frequencyscore(w)$ is the frequency of the term w in the set of $k_{\mathrm{d}}^{(freq)}$ documents returned by the initial query. The frequency score is normalized by by dividing it by the maximum frequency value which is the maximum of the frequencies of the candidate terms chosen from $k_{\mathrm{d}}^{(freq)}$ documents. α is a tuning parameter that controls the trade-off between the contextual score and the frequency score. Its value is set to a value that lies between 0 to 1(The exact value of α chosen for implementing the proposed model is experimentally determined).

After ranking the candidate expansion terms using Equation (4.6), we choose the top $\mathrm{m_q}$ terms($\mathrm{m_q}$ is tuned for the best results). The selected expansion terms are added to the initial query terms to form an expanded query.

Expanded Query = Original Query ∪ Selected Expansion Terms

4.2.5 Performing Search Using Expanded Query

We get the final search results by submitting the expanded query to the Okapi BM25 IR model. The documents returned by the expanded query are used for evaluation.

4.3 Description of the Data Set

We have tested the proposed IR model on datasets- 1) FIRE (Forum for Information Retrieval Evaluation) 2010 benchmark dataset which is mentioned as Dataset-1 in the rest of the paper. The description of the dataset is already described in section 2.4.

4.4 Evaluation, Experiments, and Results

To evaluate retrieval models, the retrieval models have been tested for 19 queries to search relevant documents from a corpus of approximately 3255 documents. To prove the generalization capability of the proposed models, the proposed retrieval models have been tested on Dataset-1(FIRE dataset) which is described in the earlier section 2.4.

4.4.1 Evaluation

To evaluate the retrieval model's effectiveness we have used a well-known technique mean average precision(MAP) which is already described in the previous section 1.6.1.

4.4.2 Experiments

We have conducted seven experiments to find the best perming IR model on the FIRE 2010 dataset. For this purpose, seven models are developed. One model differs from another in expansion term selection methods. The models are developed by considering all possible combinations of the three expansion term extraction methods, (1) expansion term extraction using WordNet,(2) expansion term extraction using word embedding, and (3) expansion term extraction using term frequency. The main objective of developing such models is to judge the efficacy of individual term expansion methods and their various combinations and to compare the proposed hybrid model with its possible variants.

Our developed seven IR models are as follows :

- Model A: It uses query expansion using a synonym set. In this case, the set of synonyms for the original query words are considered as the candidate expansion terms. WordNet is employed for obtaining the synonyms. Finally, after ranking the candidate terms using Equation (4.6), m_{q} terms are selected for query expansion.

- Model B: It uses query expansion using only the semantic words that are semantically similar to the original query words. The word embeddings are used for finding semantically similar words from the top $k_{\mathrm{d}}^{(emb)}$ documents retrieved by the initial query. The words whose word embedding-based cosine similarities with the query words pass a predefined threshold value(threshold value of 0.7) are added to the set of candidate expansion terms. Finally, after ranking the candidate terms using Equation (4.6), m_{q} terms are selected for query expansion.

- Model C: It uses query expansion using the frequent set of words. For this purpose, the top 30 frequently occurring words are chosen from the top $k_{\mathrm{d}}^{(freq)}$ documents as the candidate expansion terms. Finally, after ranking the candidate terms using Equation (4.6), m_{q} (set as 5) terms are selected for query expansion.

- Model D: It uses the first hybrid query expansion model that combines the synonym set(same as Model A) and the most frequent set(same as Model C) to create a pool of candidate expansion terms. Finally, after ranking the candidate terms using Equation (4.6), m_{q} terms are selected for query expansion.

- Model E: It uses the second hybrid query expansion model that combines the synonym set (same as Model A) and the semantic set (same as Model B) to create a pool of candidate expansion terms. Finally, after ranking the candidate terms using Equation (4.6), m_{q} terms are selected for query expansion.

- Model F: It uses the third hybrid query expansion model that combines the semantic set (same as Model B) and the most frequent set (same as Model C) to create a pool of candidate expansion terms. Finally, after ranking the candidate terms using Equation (4.6), m_{q} terms are selected for query expansion.

- Model G: It uses the proposed hybrid model that combines all three candidate expansion term extraction methods. In this case, the synonym set (same as Model A), the semantic set (same as Model B), and the most frequent set (same as Model C) are combined to create a pool of candidate expansion terms. Finally, after ranking the candidate terms using Equation (4.6), m_{q} terms are selected for query expansion.

4.4.3 Parameter tuning

In this subsection, we present how the optimal values for the model parameters are obtained.

Table 4.1: The best parameter values set for three base models, Model A(Synonym-set), Model B(Word embedding-based) and Model C(Frequency-based)

Model Name	Parameters
`Model A (Synonym-set)`	α=0.9, m_{q}=5,
`Model B (Word embedding-based)`	$k_{\mathrm{d}}^{(emb)}$ = 10, α=0.8, m_{q}=5
`Model C (Frequency-based)`	$k_{\mathrm{d}}^{(freq)}$ = 5, α=0.9, m_{q}=5

For each of the above-mentioned models developed by us, there are several tuning parameters that need to be tuned to achieve better retrieval performance. These important tuning parameters are:- (1) k which indicates the number of documents chosen from the initial search results returned by the original query. k has been indicated by $\mathrm{k}_{\mathrm{d}}^{(freq)}$ where the frequency-based candidate term extraction method is used, but it is indicated by $\mathrm{k}_{\mathrm{d}}^{(emb)}$ where the embedding-based candidate term extraction method is used, (2) α, which is a blending parameter used in Equation (4.6), and (3) m_{q} that indicates the number of expansion terms finally selected for query expansion. We have observed that the best possible values of these parameters vary from one model to another model.

For Model A there are only two tuning parameters α and m_{q}. For the best results, we perform a grid search on the values of α and m_{q} where α is varied from 0 to 1 with an interval of 0.1 and m_{q} is varied from 5 to 40 with an interval of 5. Therefore, we consider a total of 88 combinations of α and m_{q} and finally choose their values that give the best MAP score for our dataset.

For Model B, other than α and m_{q}, there is another parameter $\mathrm{k}_{\mathrm{d}}^{(emb)}$ that indicates the number of documents chosen from the initial search results. To achieve the best retrieval performance, we tune $\mathrm{k}_{\mathrm{d}}^{(emb)}$ by varying its value from 5 to 20 with an interval of 5. For each value of $\mathrm{k}_{\mathrm{d}}^{(emb)}$, a grid search is performed on the values of α and m_{q} by varying α from 0 to 1 with an interval of 0.1 and m_{q} from 5 to 40 with an interval of 5. Thus for each value of $\mathrm{k}_{\mathrm{d}}^{(emb)}$, we consider a total of 88 combinations of α and m_{q} and finally choose their values that give the best MAP score for our dataset. Using the similar method, the parameters $\mathrm{k}_{\mathrm{d}}^{(freq)}$, α and m_{q} are tuned for Model C. The parameter values for which Model

A, Model B, and Model C performed the best are shown in Table 4.1.

Models D to G are hybrid models, each of which uses two or three candidate expansion term extraction methods. For the hybrid models where the embedding-based term extraction method is used as a component, the value of the tuning parameter $k_{\mathrm{d}}^{(emb)}$ is set to 10 since it gives the best results for the word embedding-based model (Model B). Similarly, the value of $k_{\mathrm{d}}^{(freq)}$ is set to 5 for the hybrid models where the frequency-based term extraction method is used as a component since the frequency-based model (Model C) gives the best results with this setting.

To reduce the number of runs, for each of the hybrid models, α and $\mathrm{m_q}$ are tuned using a sequential search method. In this method, the parameters are tuned one by one. For example, for Model G (the best hybrid model). after fixing $k_{\mathrm{d}}^{(emb)}$ to 10 and $k_{\mathrm{d}}^{(freq)}$ to 5, α is varied from 0 to 1 with an interval of 0.1 to find the optimal value for α. Then fixing the values of $k_{\mathrm{d}}^{(emb)}$, $k_{\mathrm{d}}^{(freq)}$, and α to their optimal values, m_{q} is varied from 5 to 40 with an interval of 5. Figures 4.3 to 4.6 show how the MAP score for each hybrid model is affected when an individual parameter is varied when the other parameters are fixed to the best possible values.

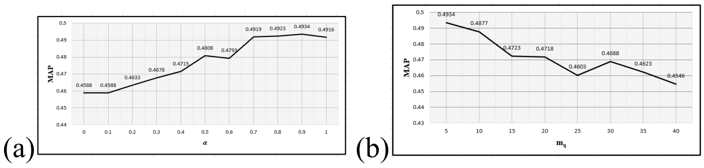


Figure 4.2: (a)Retrieval performance of Model D (Synonym-set+Frequent-set) when α is varied (b) Retrieval performance of Model D (Synonym-set+Frequent-set) when m_{q} is varied

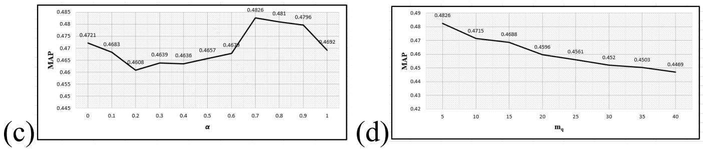


Figure 4.3: (c) Retrieval performance of Model E (Synonym-set + Semantic-set) when α is varied (d) Retrieval performance of Model E (Synonym set+Semantic-set) when m_{q} is varied

The values of the parameters for which the hybrid models gave the best results are reported in Table 4.2.

4.4.4 Results

Table 4.3 displays the MAP scores achieved by the seven different IR models developed by us.

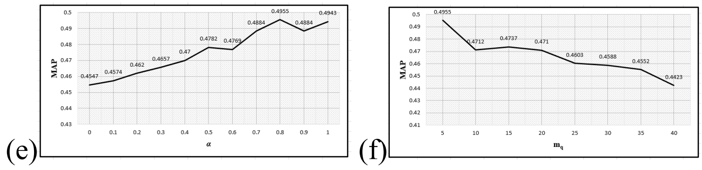


Figure 4.4: (e) Retrieval performance of Model F (Semantic-set + Frequent-set) when α is varied (f) Retrieval performance of Model F (Semantic-set+Frequent-set) when m_{q} is varied

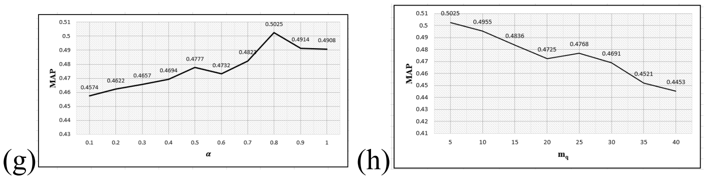


Figure 4.5: (g) Retrieval performance of Model G (Synonym-set + Semantic-set + Frequent-set) when α is varied (h) Retrieval performance of Model G (Synonym-set + Semantic-set + Frequent-set) when m_{q} is varied

The results shown in Table 4.3 demonstrate that Model G combining a synonym set, semantic set, and the most frequent set for expansion term extraction outperforms other alternative models (Model A-F) developed by us. Therefore it shows that the combination of three expansion term extraction methods is useful for achieving the best performance. We can also see from the table that, among the individual expansion methods, the frequency-based method is the best, but all the hybrid models except Model E perform better than the models that use a single expansion method. These results suggest that the hybridization of various expansion methods is useful in improving IR performance.

4.4.5 Comparison with the BM25 IR model

We have used BM25 as the core component with which the proposed hybrid query expansion method is combined for implementing our models. To prove the effectiveness of the proposed models, we should compare their performance with BM25 without query expansion.

In the BM25 model, the parameters $K3$, B, and $K1$ are tuned using a sequential search method. For example, for Model BM25 after fixing $K1$ to 2.2 and B to 0.3, $K3$ is varied repeatedly from 50, 100, 150, 200, 250, and 300. with an interval of 50 to find the optimal value for $K3$. Then fixing the values of $K3$, $K1$, and B is varied from 0 to 1 with an interval of 0.1 to find the optimal value for B. B to their optimal values, $K1$ is varied from 0.2 to 3 with an interval of 0.2. Figure 7 shows how the MAP score of the BM25 model is affected when an individual parameter is varied and the other parameters are fixed to the width=15cm

Table 4.2: The parameter settings producing the best MAP scores for various hybrid models: Model D(Synonym-set + Frequent-set), Model E(Synonym set + Semantic-set), Model F(Semantic-set+Frequent-set), and Model G(Synonym-set +Semantic-set+Frequent-set) .

Model Name	Parameters
Model D(Synonym-set+Frequent-set)	$k_{\mathrm{d}}^{(freq)} = 5$, α=0.9, m_{q}=5
Model E(Synonym-set+Semantic-set)	$k_{\mathrm{d}}^{(emb)} = 10$, α=0.7, m_{q}=5
Model F(Semantic-set+Frequent-set)	$k_{\mathrm{d}}^{(freq)} = 5$, $k_{\mathrm{d}}^{(emb)} = 10$, α=0.8, m_{q}=5
Model G(Synonym-set+Semantic-set +Frequent-set)	$k_{\mathrm{d}}^{(freq)} = 5$, $k_{\mathrm{d}}^{(emb)} = 10$, α=0.8, m_{q}=5

width=9cm

Table 4.3: Performance Comparisons of various IR models on the FIRE dataset

Model Name	MAP Score
Model A(Synonym-set)	0.4796
Model B(Word embedding-based)	0.4896
Model C(Frequency-based)	0.4923
Model D(Synonym-set+Frequent-set)	0.4934
Model E(Synonym-set+Semantic-set)	0.4826
Model F(Semantic-set+Frequent-set)	0.4955
Model G(Synonym-set+Semantic-set +Frequent-set)	0.5025

best possible values.

Table 4.4 compares the proposed best-performing model (Model G) with the BM25 model without query expansion.

Table 4.4: Performance comparisons of the proposed best-performing model (Model G) with BM25 without query expansion

Model Name	MAP
BM25 without query expansion	0.4863
Model G (Synonym-set + Semantic-set + Frequent-set)	0.5025

After analyzing the results in Table 4.4, we can see that Model G(Synonym-set + Semantic-set + Frequent-set) gives us the best MAP score of 0.5025 for the FIRE 2010 dataset whereas the BM25 model gives us the MAP score of 0.4863, that is, the proposed hybrid query expansion-based IR model outperforms the BM25 model without query expansion by a wide margin.

By comparing Table 4.3 and Table 4.4, we can infer that Model B(Word embedding-based) and Model C(Frequency-based) each individually outperform the scores of the BM25 Model without query expansion whereas Model A(Synonym-set) individually does not. Among the hybrid models, Model D(Synonym-set + Frequent-set) and Model F(Semantic-

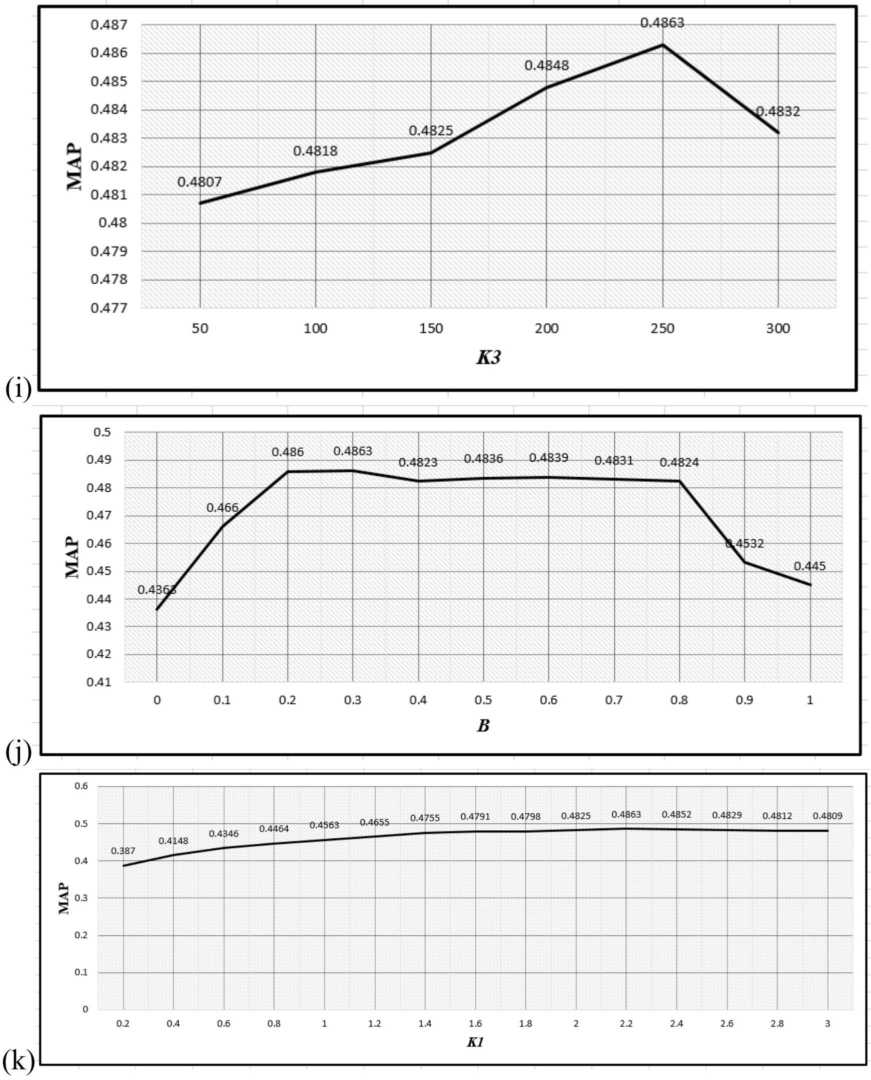


Figure 4.6: (i) Retrieval performance of model BM25 when K3 is varied (j) Retrieval performance of model BM25 when B is varied (k)Retrieval performance of model BM25 when K1 is varied

set + Frequent-set) perform better than the BM25 Model without query expansion, but the MAP score for Model E(Synonym-set + Semantic-set) is relatively poor. For Model E, it is apparent that incorporating the synonym set cannot complement the semantic set created using word embedding and hampers the performance. After analyzing the results, we can conclude that the hybrid models: Model A(Synonym-set), Model B(Word embedding-based), and Model C(Frequency-based), Model G(Synonym-set + Semantic-set + Frequent-set), outperform the BM25 without query expansion by a wide margin.

Therefore these results prove that the hybrid query expansion method is effective if the

Table 4.5: Percentage improvement achieved by the proposed models over the BM25 model without query expansion

Model Name	Improvement
Model A(Synonym-set)	-1.37%
Model B(Word embedding-based)	+0.67%
Model C(Frequency-based)	+1.23%
Model D(Synonym-set+Frequent-set)	+1.46%
Model E(Synonym-set+Semantic-set)	-0.76%
Model F(Semantic-set+Frequent-set)	+1.89%
Model G(Synonym-set+Semantic-set +Frequent-set)	+3.33%

proper experimental setup and parameter tuning are used.

$$Performance\ Improvement(PI) = \frac{y-z}{z} \times 100 \tag{4.7}$$

Where:
y=MAP Score obtained by the Proposed approach
z=MAP Score obtained by baseline approach

Using Equation (4.7), we have calculated the percentage improvement in the performance of our proposed models over the BM25 model without query expansion, and the percentage improvement results are presented in Table 4.5.

The performance improvement results shown in Table 4.5 indicate that models B, C, D, F, and G outperformed the BM25 model without query expansion. Compared to the BM25 model without query expansion, models B, C, D, F, and G achieve 0.67%, 1.23%, 1.46%, 1.89%, and 3.33% improvement, respectively. We have obtained negative results for Model A (Synonym-set) and Model E (Synonym-set+Semantic-set). For both cases, the performance is slightly degraded. These results are similar to some previous works found in the literature that also reported negative results for query expansion using synonyms[110, 111].

The fact that our proposed model G outperformed BM25 by more than 3% suggests that the final term selection process using three sets of candidate term extraction methods is most effective for Bengali information retrieval.

4.5 Chapter Summary

The automatic query expansion technique is commonly used for dealing with the vocabulary mismatch problem in information retrieval tasks. To deal with the vocabulary mismatch problem, we propose a hybrid query expansion (QE) method in this chapter. The proposed method has two phases- (1) the candidate expansion term generation phase and (2) the expansion term selection phase. In the candidate expansion term generation phase, the statistical, lexical, and semantic methods are combined to automatically generate the candidate expansion terms that have relations with a given query. The candidate expansion terms extracted by these three methods are combined to create a pool that is further processed to choose the final expansion terms. In the expansion term selection phase, the candidate terms are ranked based on their contextual and semantic relatedness with the given query. To prove the effectiveness of the proposed hybrid query expansion method, we incorporate the proposed method in the Okapi BM25 IR model and then we test the retrieval performance of the enhanced Okapi BM25 IR model on the FIRE dataset which is a benchmark dataset for Bengali information retrieval. We observed the following hypothesis from this chapter: (1) the hybrid query expansion method is more effective than an individual query expansion method for Bengali information retrieval (2) the synonym-based query expansion is not alone effective for Bengali information, but it is useful when it is combined with other query expansion methods (3) the experimental results reveal that the hybrid approach that uses WordNet-based, Word embedding-based, and frequency-based term extraction methods are superior to other hybrid models developed by us.

Although the proposed model has been tested on a benchmark Bengali IR dataset, it can be ported to other language domains with minor modifications.

Our next plan is to design a hybrid model IR model by combining two or more IR models.

5 Bengali Information Retrieval Using Model Combination

Any IR model comes with a set of assumptions and each model fails under different circumstances. When one model produces poor IR performance for some queries, there may be another IR model that is more accurate on those queries. Therefore, by suitably combining multiple IR models, overall retrieval performance can be improved if they complement each other.

We have observed that the Latent Semantic Indexing (LSI)–based IR model presented in Chapter 3 maps documents and queries to a new reduced space called latent semantic space and it computes the similarities between the documents and the queries in the latent semantic space. This approach improves recall of the IR system [28], but it also hampers precision because the semantic model does not capture the relative importance of query words present in the documents.

Another semantic IR model presented in Chapter 3 uses fixed word embedding for document and query representation. This model also helps to improve recall, but it may hamper precision. It happens when the corpus size is not too large.

The TFIDF-based vector space model [10, 8] is a popular IR model widely used by many researchers for information retrieval tasks. Though it suffers from the word mismatch problem or vocabulary mismatch problem, it has also some advantages. The main benefit of the traditional vector space model [10, 8] is that it can capture the relative importance of the terms in the documents and queries since it represents queries and documents as the vectors of the TF*IDF weights of the terms, and thus it models effectively the dissimilarity among the documents through data sparsity. Though the TF-IDF-based vector space model gives poor recall due to vocabulary mismatch problems, it can hold control on precision.

Therefore, we hypothesize that the outputs of the two models -(1) the traditional TFIDF-based vector space model and (2) the semantic-based IR models like the LSI-based IR model or word embedding-based IR model can be combined using proper blending functions. Tuning the combination parameters appropriately can yield a new hybrid model that can improve information retrieval performance.

In this chapter, we propose two hybrid IR models that are developed using model combination, (1) Hybrid model-1 which combines a TFIDF-based vector space model(VSM) with the latent semantic indexing(LSI) model, and (2) Hybrid model-2 which combines a TFIDF-based vector space model (VSM) with the word embedding-based model.

In the first part of this chapter, we discuss how the queries and documents are preprocessed. In the second part of this chapter, we describe the base IR models such as the TF-IDF-based IR model, LSI-based IR model, and Word Embedding-based IR model which are used to develop the hybrid models. In the third part of the chapter, we describe in detail how the base models are combined to build the hybrid models.

The rest of the paper is organized as follows. In section 5.4, we evaluate our proposed models using a benchmark dataset. Experiments on the benchmark data set show that our proposed technique achieves state-of-the-art performance for Bengali information retrieval. Finally, we conclude in Section 5.5. In this section, we also suggest some future work for further enhancement of the proposed technique.

5.1 Methodology

In the previous chapters of this thesis, we presented single IR models like the TFIDF-based vector space model, LSI-based model, or word embedding-based model. Comparisons of the single IR models reveal that there exists no single IR model that performs consistently well for all queries. We observe that, when one model produces better IR performance for some queries, another IR model produces relatively poor performance for those queries. Therefore, to develop a more robust IR system, we combine multiple IR models to exploit the strengths of the individual models for building a hybrid model that can be more accurate than an individual base IR model.

In this chapter, we proposed two hybrid IR models:

- Hybrid model-1, which combines a TFIDF-based vector space model(VSM) with the latent semantic indexing(LSI) model.
- Hybrid model-2, which combines a TFIDF-based vector space model (VSM) with the word embedding-based model.

An architecture of the first Hybrid model-1 (Model-1) is shown in Figure 5.1. In this case, two different IR models, TFIDF-based VSM and LSI-based model, accept the same input query and then the outputs of these models are linearly combined to assign a unique relevance score to each document with respect to the given query. Since the output of an individual model is produced in the form of similarity values indicating the degree of relevance of a document with the query, the combined model gives hybrid similarity scores. Documents relevant to the query are then ranked based on the combined scores.

Similarly, the architecture of the second Hybrid model-2 is shown in Figure 5.2. The main difference between this model from the first combined model is that this model uses a word embedding-based model instead of the LSI-based model. In this case, the word

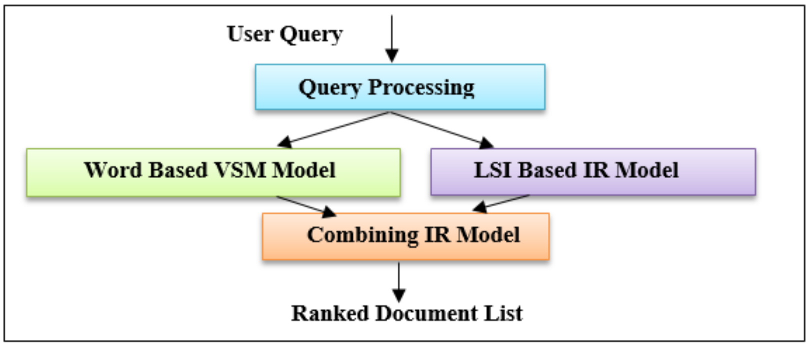


Figure 5.1: Hybrid model-1 Architecture

embedding-based model is combined with TFIDF-based VSM. Hybrid score calculation and ranking procedures are the same as the first model.

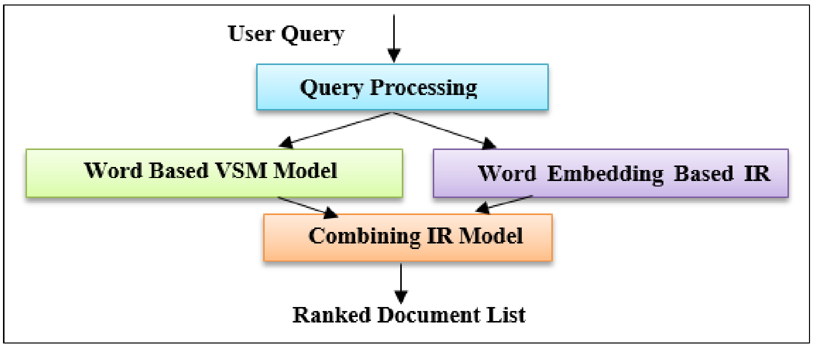


Figure 5.2: Hybrid model-2 Architecture

In the subsequent subsections, we briefly present the three different single IR models that take part in building the two different combined models. Then we discuss how the output scores of the single IR models are combined.

5.1.1 TFIDF-based Vector Space Model for IR

This model was widely used for document retrieval [10, 1]. In this model, both the documents and queries are mapped to an n-dimensional vector space where n is the size of the vocabulary created from the corpus, and a dimension corresponds to a vocabulary word(term) (see Figure 1.3). In this space, a component of a vector is the TF-IDF value of the corresponding vocabulary word. The relevancy of a document with a given query is calculated using cosine similarity between their vector representations. In contrast to the earlier Boolean IR model, this model captures the relative importance of the terms present in the document. A basic architecture in Figure 5.3, the TFIDF-based vector space model has several components such as document representation, query processing, query representation, similarity measure, and ranking.

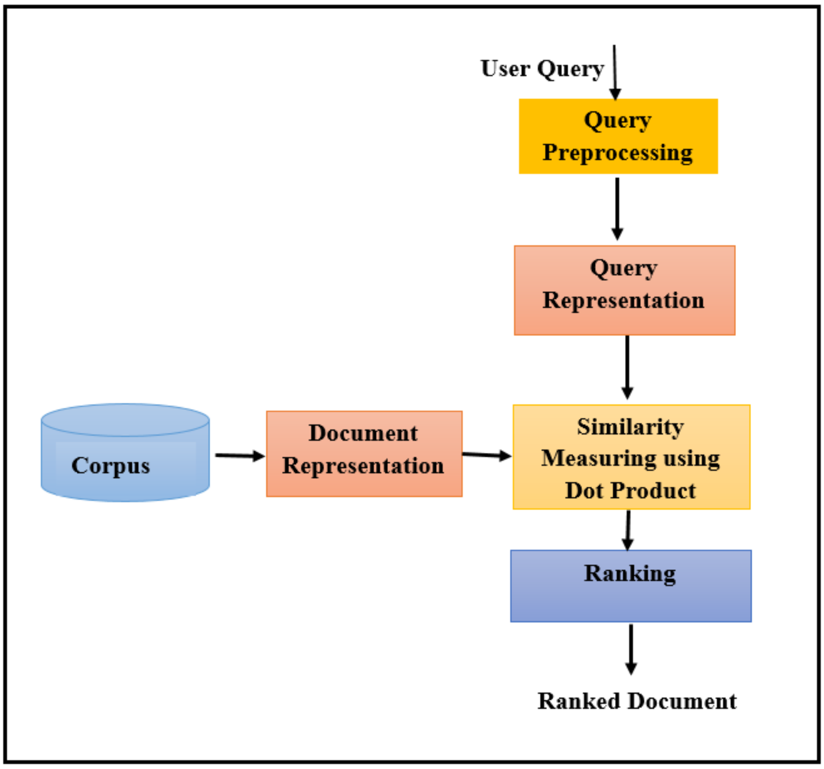


Figure 5.3: Vector Space Model Architecture

Document Representation

Document Representation involves three primary steps: tokenization, Stemming /Stop word removal, and storing the information on file with a special data structure for fast access during query processing and vector representation. First punctuation was removed from all documents. Stop-words were then removed using the list of stop-words provided by FIRE. The documents are then tokenized into a collection of words using the so-called Bag-of-words model. Stemming was performed next using the Yet Another Suffix Stripper (YASS)[6]. We have built the inverted index by sort-based indexing for improving retrieval performance[4].

The Bag-of-Words model [10], views a document/a query as a collection of words. Given a collection of documents C, containing words from a vocabulary V, the following information can be extracted from each document.

Term Frequency (TF): For a word, the Term Frequency measures the frequency of a word in the document. We have used a modified form of *TF* as:

$$Modified\_TF = log(0.5 + TF) \tag{5.1}$$

Inverse Document Frequency (IDF): The Inverse Document Frequency (IDF) of the word q_i is calculated in terms of DF as:

$$IDF(q_i) = log[0.5 + \frac{N}{DF(q_i)}] \tag{5.2}$$

Where: N is the corpus size.
DF is the count of documents containing the word at least once. It is computed considering the entire collection of N documents.

Vector Representation: As Bag-of-Words model represents each document and query as collection of words, each document and query is represented as a vector of length v is the vocabulary size. Each component of the document vector or query vector corresponds to a word in v. A document vector for d is $\vec{d} = (w_1 , w_2 , \ldots w_v)$, where w_i is TF*IDF weight of word x with the ith index in the vocabulary and x is present in d. If the x is not present in d, w_i is set to 0. TF and IDF is calculated using Equation 5.1 and Equation 5.2 respectively.

Query Pre-processing

When a user gives the query in the system, the system passes it into the retrieval engine. The retrieval engine passes the query to the query processing phase which processes the query in the same way as the document is processed. Tokenization, stop word removal, and stemming are applied to the query also. After the initial processing of the query, it is passed to a vector space model which matches a query vector to the document vectors for ranking.

Query Representation

For a given query q, we compute the query vector $\vec{q} = (q_1 , q_2 , \ldots q_v)$ where q_i is TF*IDF weight of the i-th vocabulary word present in query q. In this case, TF indicates the frequency of query words in the query q.

Similarity Measuring using Dot Product

For the sake of computational efficiency, we use dot product [10] of the document vector and query vector as the relevance score instead of computing cosine similarity. The relevance score for a document d is:

$$TF - IDF\_Score(d, q) = \sum_{w \in q \cap u \in d} TF - IDF(w) \times TF - IDF(u) \tag{5.3}$$

Since the dot product becomes too large and the output of this model is finally combined with another IR model, the model's output should be properly normalized. We apply the softmax function to normalize the relevance scores assigned by this model as follows:

$$Softmax\_normalize\_value(d, q) = \frac{e^{Modified\_TF - IDF\_Score(d,q)}}{\sum_{d \in D} e^{Modified\_TF - IDF\_Score(d,q)}} \tag{5.4}$$

where:

$$Modified\_TF - IDF\_Score(d, q) = \log(TF - IDFScore(d, q)) \tag{5.5}$$

The result of equation (5.4) is also very small. So, we normalize again this value between 0 and 1 by using the traditional min-max procedure. And after this min-max normalization, we call it as RS1, which is used while combining IR models (presented in the later section). For the sake of computational efficiency, for each query, we consult an inverted index to retrieve documents relevant to the query words at a time. While information of the relevant documents is extracted from the inverted index, TF-IDF information is also extracted.

5.1.2 Ranking Documents

For a given query and each document in the corpus, computed dot product between the corresponding query vector and the document vector. Then rank these documents are based on the cosine similarity score.

5.2 Latent Semantic Analysis (LSA) approach for Bengali IR

Latent Semantic Analysis (LSA) is an algebraic-statistical technique for representing meanings of words by their contextual usages and mapping documents into low-dimensional abstract concept space where a concept is represented by the set of words appearing in similar contextual usages. The similarities between a query and the documents are computed using cosine similarity between their vector representation in the latent semantic space of a chosen dimension. The cosine similarity scores between a query and the documents are considered as the relevance scores of the documents with the query. The details of this model are already discussed in Chapter 3.

5.3 Word embedding-based model

Word embedding(WE) is a popular and effective method [29, 36] for word representation and finding the semantic similarity between words. Technically, it represents a word as a vector and it maps the words which are contextually similar nearby in the embedding space. In this model, a document or query is represented as a vector using the vectors of the words contained in the document or query. Thereafter, the cosine similarities between a query and the documents are computed to obtain the relevance scores of the documents with the query. The details of this are already discussed in Chapter 3.

5.3.1 Combining IR Models

We combine the literal term matching method with the semantic-based approach to blend the benefits of the two IR approaches.

1. One is, combines a traditional TFIDF-based vector space IR model with the word embedding-based IR model whose method name is Combine model-1

2. Another is, to combine a traditional TFIDF-based vector space IR model with the LSI-based IR model whose method name is Hybrid Model-2

Word embedding-based IR model, and LSI-based IR model are already discussed in Chapter 3.

Hybrid model-1

We design a hybrid model to make use of the benefits of the traditional TFIDF-based vector space IR model and the LSI-based model. The traditional TFIDF-based vector space IR model can discriminate among the documents using the term frequencies measuring relative term importance, but this model's performance is affected by the word mismatch problem. On the other side, the LSI-based model uses semantic matching and alleviates the word mismatch problem. Although the LSI-based model improves the recall value, it exhibits poor precision. So, combining the outputs of the two models can complement each other. As we mentioned in earlier sections, both IR models give relevance scores for each document. If RS1 and RS2 are the relevance scores outputted by traditional TFIDF-based vector space IR model named Model 3 and the LSI-based model named Model 1 respectively for a document, the hybrid relevance score for the document is obtained using Equation (5.6).

$$RS = \beta\dot{R}S1 + (1-\beta)\dot{R}S2 \tag{5.6}$$

After ranking all documents using their relevance scores calculated using Equation (5.6), the hybrid model returns the top M documents.

Hybrid Model-2

As we mentioned earlier, the traditional TFIDF-based vector space IR model has the advantage of discriminating terms based on their frequency-based relative importance and preserving dissimilarity between the documents based on data sparsity. On the other hand, this model suffers from a word mismatch problem which can be tackled with a word embedding-based IR model. So, we combine the outputs produced by the two models to blend the benefits of both models to improve retrieval accuracy. As we discussed earlier in this chapter, both models give outputs in terms of similarity scores computed by comparing a given query and documents in the corpus. For a given query q submitted to both the models, each model assigns a degree of relevance based on its similarity with documents in the corpus. Thus, each document in the corpus can get two relevance scores assigned by two different models with respect to a given query. If the relevance scores assigned by Model 1 and Model 2 for a document are RS1 and RS3 respectively, the combined relevance score for the document with respect to a query is obtained using the following equation.

$$RS = \alpha\dot{R}S1 + (1 - \alpha)\dot{R}S3 \tag{5.7}$$

where α is the blending parameter indicating the weight assigned to the first IR model. After re-computing the relevance score of each document using Equation (5.7), documents are re-ranked. Finally, top m documents are returned by the hybrid model. Here the value of α is determined through our experimentation on our data sets and FIRE dataset.

5.4 Evaluation, Experiments, and Results

The two hybrid models proposed in this chapter are tested on the following two Bengali datasets.

- Dataset used in [1], contains 19 queries to search relevant documents from a corpus of approximately 3255 documents. The details of this dataset are discussed in Chapter 3.

- FIRE Dataset (which was mentioned as Bengali Dataset-1 in Chapter 2) contains 123021 documents and a relevance file. The relevance file contains 50 queries(query numbers 76 to 125) and the human relevance judgment for each query.

5.4.1 Evaluation

To evaluate the retrieval model's effectiveness, we used a well-known technique mean average precision(MAP), which is already described in the previous section 1.6.1.

Expriment 1

For this experiment, firstly, we have implemented the proposed traditional TFIDF-based vector space IR model(Model 3) [1] then implemented the proposed latent semantic indexing(LSI) based IR model(Model 1)[1] to Bengali information retrieval separately. Then compute the cosine similarity between the document and query for each model. Since k is the most important parameter of the LSI-based model and it indicates the dimension of semantic space into which documents and queries are mapped, we have tuned this parameter to achieve better results. The effect of varying k values on the MAP score for Model 1 is shown in Figure 5.4. It indicates that Model 3 with k set to 95 gives the best MAP score. Then hybrid relevance score for the document is obtained using Equation (5.6).

Experiment 2

In this experiment, we used the word embedding-based model presented by Mikolov et al. [29] to compute the semantic similarity between the query and a document based on the semantic similarity between the query words and document words. We have used the gensim word2vec model. Which gives the word vector of length 300. We locally compute the similarity between a document and a query. For computing similarity between documents and a query, both documents and queries are represented as numeric vectors. Which is already described in section 3.2.3. The word embedding model gives the best results when the word similarity threshold is set to 0.9. After re-computing the relevance score of each document using Equation (5.7), documents are re-ranked. Finally, top m documents are returned by the hybrid model.

5.4.2 Results and Discussion

Table 5.1 displays the MAP scores obtained by our two combined IR models developed by us on Dataset-3 which is our dataset.

Table 5.1: Performance Comparisons of the proposed two IR models on our dataset[1]

Model Name	MAP Score
The proposed Hybrid model-1 [33]	0.5960
The proposed Hybrid model-2 [1]	0.5805

Table 5.2 shows the MAP scores obtained by the variants of our proposed three IR models for Dataset-1 which is the benchmark dataset. The variants of our proposed model are (1) the Hybrid model-1, and (2) the Hybrid Model-2 for the IR model. The results shown in Table 3.2 reveal that our proposed model Hybrid model-1 performs better than the Hybrid model-2. This also shows the generalization capability of the proposed model. Table 5.2 indicates that our proposed combined model is effective for Bengali information retrieval tasks.

Table 5.2: Performance Comparisons of the proposed three IR models on the Dataset-1 [1]

Model Name	MAP Score
The proposed Hybrid model-1 [33]	0.4535
The proposed Hybrid model-2[1]	0.4332

Parameter tuning

In this subsection, we present how we obtain the optimal values for the model parameters. We observed that the most important parameter whose values affect the retrieval performance of Hybrid model-1 and Hybrid model-2 are α and β. Since the combined model is a hybrid model of two other models and it computes the relevance of a document with respect to a query based on the weighted combination of the relevance scores given by the two different component models, how much weight should be assigned to the model components' output is an important parameter to be tuned for better results.

For the proposed Hybrid model-1, we have an important tuneable parameter β which determines how much weight should be assigned to an individual component model's output for achieving a better MAP score. Fig.5.3 shows the impact of β on this model. It indicates that the model with β set to 0.492 gives the best MAP score. We have shown in Figure 5.4 the effect of the blending parameter (weight) on the performance of our proposed hybrid model. It is evident from Figure 5.4 that we get the best results on our data set when the value of the blending parameter α is set to 0.23.

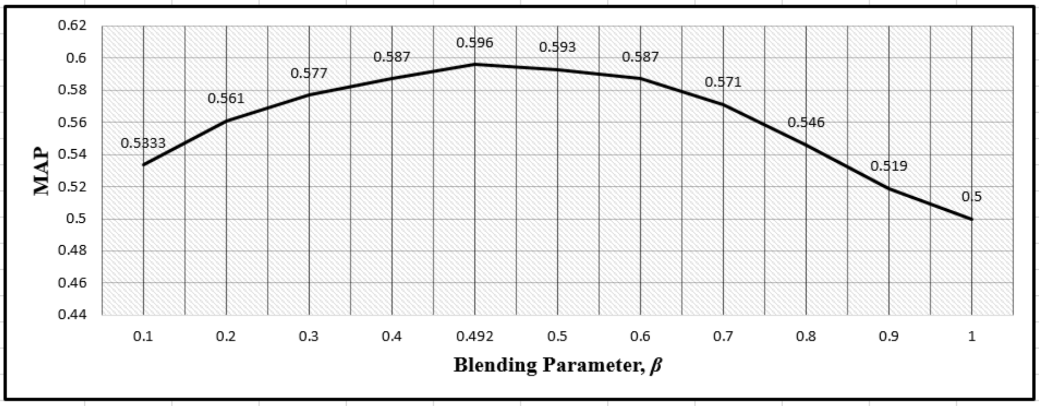


Figure 5.4: Impact of β on the proposed Hybrid model-1 when β is varied on the Dataset-3

Analysis and Discussion

The word Embedding based model individually cannot outperform the traditional TFIDF-based VSM, but when they are combined the performance improves. The reasons that we

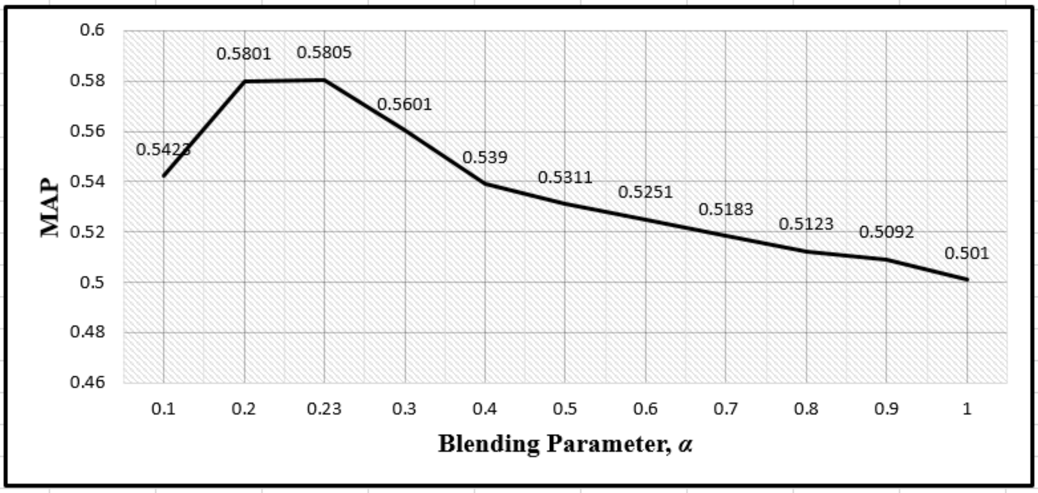


Figure 5.5: Impact of blending parameter α on Hybrid model-2 when α is varied on the Dataset-3

identify are:

- Unlike the TFIDF-based VSM model which shows very low similarity scores between queries and documents for many cases, the word embedding-based model shows relatively high similarity values for those cases because it can capture the semantic similarity between queries and documents. However, the word embedding-based model is not always beneficial due to false positive and false negative issues. Moreover, the Word embedding-based model does not take into account the relative importance of terms in documents and queries.

- Through word embedding-based IR model can capture the semantic relationship between query and documents, it tends to assign relevance scores to many documents in the set. On the other hand, the TFIDF-based VSM model considers the relative importance of query words present in documents as well as data sparsity which adds dissimilarity among the documents to the ranking procedure. Thus, the word embedding-based model attempts to improve recall whereas the TFIDF-based vector space model attempts to improve precision. Finally, the effect of hybridization leads to better information retrieval performance.

- Though the LSI-based model can do semantic matching, it tends to retrieve too many documents relevant to the query. However, when it is combined with the VSM model, the overall retrieval performance is improved. This is because the VSM model can discriminate among the documents retrieved by the LSI-based model using the relative importance of query words present in the documents. In this way, they complement each other when both models are combined.

5.5 Chapter Summary

In this chapter, we present a hybrid IR model that combines the relevance scores produced by two different individual IR models for each document-query pair and ranks documents based on the combined scores. We hybrid various IR models to take advantage of the strengths of the individual models in order to create a hybrid model that is more accurate than the individual base model. So, we integrate the literal term matching method with the semantic-based approach to combine the benefits of the two IR approaches. In this chapter, we proposed two hybrid IR models: (1) Hybrid model-1 which combines a TFIDF-based vector space model(VSM) with the latent semantic indexing(LSI) model.

(2) Hybrid model-2 which combines a TFIDF-based vector space model (VSM) with the word embedding-based model. Our experiments show that our proposed hybrid models perform better than the individual component model.

Although the proposed model has been tested on a benchmark Bengali IR dataset, it can be ported to other language domains with minor modifications. The experimental results show that our proposed model for Bengali document retrieval is effective and it outperforms several baseline IR models, like the TFIDF-based vector space model(VSM), latent semantic indexing(LSI) model, and the word embedding-based model. One important factor for the IR task is to make the retrieval process faster. When the model combination is made, the hybrid system becomes relatively slow.

Our next plan is to design a Multi-document summarization(MDS) model.

6 Clustering and Summarizing Search Results

6.1 Introduction

Internet users are overwhelmed by a vast amount of information when they place a search query using a search engine because the search engine returns thousands of text documents in response to a single query. The common practice that the user follows is to check the first 10-20 hits and discard the remaining documents if they do not find the relevant documents within the top-ranked documents. In this case, the users may miss the relevant documents having higher rank positions in the ranked list. This leads the users to reformulate the query and repeat the process. Thus it takes a long time to find relevant information on the Internet. This is known as the information overload problem. To overcome this problem, the summarization tools can be useful for summarizing the search results returned by the search engine and presenting initially the summary of the related search hits to the user.

In this chapter, we propose a method that can group the search results returned by an IR model into multiple clusters using a clustering algorithm and summarize each cluster of documents using a multi-document text summarization system. For clustering search results, we have used the histogram-based clustering algorithm discussed in Chapter 2. In the summarization method, each cluster of related documents is reduced to the condensed representation which is presented to the users. The main idea is that, instead of displaying the entire cluster of documents to the users, if its gist is presented to the users, by reading the gist they can quickly find the cluster that contains the most relevant documents. When the most relevant cluster is chosen, it can be unfolded to find out the relevant information. Thus it reduces the information search time [112]. For summarizing clusters of retrieved relevant documents, We propose an extractive multi-document summarization method that produces a summary by condensing a cluster of documents. Multi-document summarization(MDS) is a process of creating a single summary by analyzing a group of related documents[5].

Most existing extractive multi-document text summarization systems calculate a score of a sentence using word importance and it is further combined with a sentence position-based score, similarity to the title, etc.[77]. The common method for measuring word importance is the TF*IDF-based method [113] in which TF (term frequency) of a term (word) is calculated by counting the occurrence of the term in a document and IDF (Inverse Document Frequency) is calculated by the formula: log(M/df), where M is the size of the corpus (which may be a large collection independent from the summarization dataset) and df (document frequency) is the number of documents from the corpus that contains the term. In the TF*IDF method, TF is multiplied by IDF to obtain the TF*IDF weight for each term and this weight is used for measuring term importance.

We observe that the term importance does not only depend on its TF*IDF weight which was usually used in the existing methods. A document also contains many important terms which are not highly frequent. Motivated by this fact, we have identified various features that affect term (word) importance and trained a support vector regressor using these features for assigning a score to each term. In contrast with the existing methods that used mainly frequency statistics for measuring term importance, we focus on using a machine learning(ML)-based method for measuring term importance which is finally utilized in enhancing the performance of the MDS system. For calculating sentence scores, term weights predicted by the ML algorithm for the words contained in the sentence are summed up.

In the first part of this chapter, we discuss the pre-processing methods. In the second part of this chapter, we briefly mention how retrieved relevant documents are clustered. In the third part of this chapter, we describe the feature set and the support vector regression model that predicts word importance using the feature set designed by us. In this part, we also describe the sentence-scoring method. In the fourth part, we describe our developed Bengali multi-document summarization datasets on which the proposed model is tested. We also test the proposed summarization model on the DUC 2004 benchmark English dataset which is also described in this part. The rest of the chapter is organized as follows. In section 6.4, we present model performance on both English and Bengali datasets. Experiments on the benchmark dataset show that our proposed technique achieves state-of-the-art performance for multi-document summarization. Finally, we present a summary of this chapter in Section 6.5.

6.2 Proposed Methodology

After retrieving the documents using the IR model proposed in Chapter 4, the retrieved relevant documents are clustered and each cluster is summarized.

6.2.1 Clustering of retrieved relevant documents

For clustering the relevant documents, we have used the Histogram-based incremental clustering algorithm discussed in Chapter 2. The reason for using this clustering algorithm is that it can run in a single pass and it does not require to specify the number of clusters in advance.

6.2.2 Summarizing a cluster of documents

For summarizing a cluster of documents, we have proposed multi-document summarization method which has several steps: (1) preprocessing of documents, (2) Word importance prediction using the support vector regression model (3) sentence scoring based on word importance and sentence position, and (4) summary generation. A block diagram of the proposed summarization method is shown in Figure 6.1.

Pre-processing

The preprocessing step primarily involves breaking the input document set into a collection of sentences, stop word removal, and stemming the words. The stop word list and the stemming process vary from one language to another language.

For the English dataset, the document set is broken into sentences using the sentence tokenizer chosen from the NLTK(The Natural Language Toolkit)tool kit. The stop-words are removed from the sentences using stop word list which is available in the NLTK library. For stemming, we have used NLTK Porter Stemmer.

For the Bengali Dataset, the document set is broken into sentences using a pre-defined set of punctuation used in the Bengali language. Stop-words were removed using the list of stop-words provided by FIRE. Finally, each word is stemmed using the Bengali stemmer named Yet Another Suffix Stripper (YASS) developed by Majumder et. al.[6].

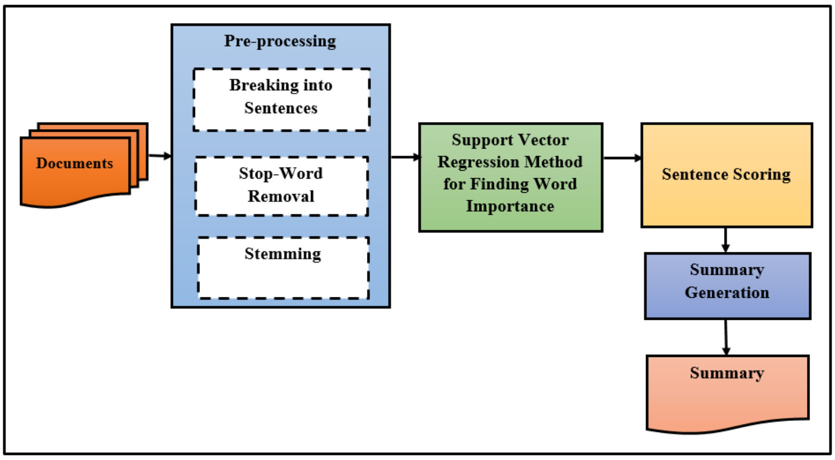


Figure 6.1: The architecture of the proposed Summarization system

[0] http://nltk.sf.net/.
[0] https://www.nltk.org/api/nltk.stem.porter.html
[0] http://fire.irsi.res.in/fire/2023/home

Word importance prediction using support vector regression model

Machine learning-based word importance predictions are the primary step of our proposed MDS system. We have used a supervised support vector regression model for predicting the importance of a term. We have used many features for training the SVR model. As we have mentioned earlier, the word importance does not only depend on its frequency. Along with the frequency value, we have used many other features such as semantic term relations, contextual information, etc. for assigning a score to each term. In this next subsection, we will discuss various features used for developing the SVR model.

Feature set

We have designed 10 features for training the SVR model. These features are as follows:

Word position in the document:

Since word importance is dependent on the word's position in the document, we consider this as a feature. The sentences of the document are numbered from 1 to n. This feature is computed as follows:

$$POS\_IN\_DOC = \frac{Positionofthesentenceinwhichthewordoccurs}{Totalnumberofsentencesinthedocument} \tag{6.1}$$

Word's position in a sentence:

We have considered two positional values for a word, one is its position in the document and another is its position in the sentence it occurs in. The Position of a word in a sentence is also considered as a feature. For this purpose, a sentence is divided into equal 3 parts- the first part, the middle part, and the last part. The occurrences of word w in the first part of a sentence are discriminated from the middle and/or the last part of the sentence in the following rules:

If w occurs within window 1 to $\frac{|s|}{3}$, then we set this feature value = 01

If w occurs within window $\frac{|s|}{3}$+1 to $\frac{|2s|}{3}$, then we set this feature value = 10

If w occurs within window $\frac{|2s|}{3}$+1 to $|s|$, then we set this feature value = 11

Where $|s|$ = length of the sentence S in terms of words *Local Term Frequency(LTF):*

A word may occur in a document frequently. The number of occurrences of a word in a document is considered as its local frequency since it is local to the document. We normalize this feature value for a word as follows:

$$LTF(w) = \frac{LTF(w)}{\max LTF(w_i)} \tag{6.2}$$

Where $LTF(w)$ = Number of times w occurs in the document and $MaxLTF(w_i)$ = Maximum LTF value in the document

Global Term Frequency (GTF):

Since we work with multi-document summarization, and the input contains multiple documents, other than the local term frequency mentioned above, the frequency of a word w in the entire input collection is also considered a separate feature. The average TF over the documents in the input cluster is taken as a feature.

$$GTF(w) = \frac{1}{|C|} \sum_{w \in C}^{|C|} TF(w) \tag{6.3}$$

Where $TF(w)$ = total count of occurrences of w in the input collection. $| C |$= the size of the input collection

TF-IDF Local:

The local term frequency (LTF) of a word is multiplied by its IDF value to define a new feature. The IDF value of the word w is computed over a corpus of N documents using equation (1.4). The value of the feature, TF-IDF Local is calculated as follows:

$$TF\_IDF\_LOCAL(w) = LTF(w) * IDF(w) \tag{6.4}$$

Where: LTF(w) = count of occurrence of the word w in a document.

TF-IDF Global:

This feature is defined by the product of GTF (defined in Equation 5.3) and IDF as follows:

$$TF\_IDF\_Global(w) = GTF(w) * IDF(w) \tag{6.5}$$

Proper Noun:

The proper nouns like organization name, person name, etc. play an important role in terms of selection. For this reason, we consider a feature that checks whether a word is a part of a proper noun or not. This is considered a binary feature.

If w is the part of a proper noun then the feature value = 1, otherwise its value = 0.

Word length(w):

Word length is considered a feature. It is observed that the words that are longer are highly informative. This is also considered a binary feature.

If length (w) > = 5 then the value of this feature =1, otherwise its value =0

Semantic frequency:

Term frequency local or global mentioned earlier is calculated by simply counting the word occurrences, which considers two words similar if they are string identical. We observed that a term can be highly similar in meaning to another word even if they are not string identical. To deal with the issue, we have designed a semantic feature called semantic term frequency. Semantic frequency is defined as a number of words in the collection to which the word w is semantically similar. The two words are said to be semantically similar if the cosine similarity between their word vectors exceeds a certain threshold (we set this threshold to 0.7). The semantic frequency of a word w is indicated by $Semantic_TF(w)$, which is calculated by counting the words to which the given word w is semantically similar. The value of this feature is computed using equation (6.6).

$$Normalized\_Semantic\_TF(w) = \frac{(Semantic\_TF(w))}{(\max Semantic\_TF(w))} \tag{6.6}$$

Where $maxSemantic\_TF(w)$ = maximum semantic TF value in the input collection

Context weight:

The importance of a word is also in some way dependent on the accompany it keeps, that is, the importance of words its surrounding words may contribute to the importance of the word under consideration. To compute this feature, a window is set up by keeping the given word at the center of the window and the words occurring in the window are collected. If the given word occurs m times, m windows are set up to extract all its surrounding words. Finally, the context weight for the given word w is calculated as follows:

$$Context\_weight(w) = \frac{1}{\mid ContextWords \mid} \sum_{w_i \in ContextWords} LTF(w_i) \tag{6.7}$$

The context weight of a word is determined by computing an average LTF of the context words occurring in the contexts of the given word. To calculate this weight, we need to set up a window in which the concerned word w is at the window center and collect words that occur within the window. The window is set up wherever in the input the word w occurs and all the context words are extracted to create a list of context words. We set the window size to 3.

Support Vector Regression (SVR) model

Support vector regression(SVR) is a regression method that is founded on the idea of Support Vector Machines (SVM). The main advantage of SVR over linear regression(LP) is that the kernel trick can be easily applied to SVR. So, we have used SVR instead of LR for word importance prediction. SVR predicts the word importance score as follows.

$$g : R^n \to R, y = g(x) =< w.x > +b \tag{6.8}$$

SVR assumes ε-cube and uses a loss function that ignores all errors inside the cube. SVR uses support vectors to improve its generalization ability. Given a training set, $< x_i, y_i >, i = 1tom$, the model parameters w and b are learned by solving the following optimization problem.

$$minimize\frac{1}{2}w^T w + c\sum_{i=1}^{m}(\xi_i + \xi_i^*) \tag{6.9}$$

Subject to

$$(< w.x_i > +b) - y_i <= \varepsilon + \xi_i \tag{6.10}$$

$$y_i - (< w.x_i > +b) <= \varepsilon + \xi_i^* \tag{6.11}$$

Where ξ_i, and ξ_i^* are the slack variables and C is the cost parameter that balances training error and model complexity.

We have used the SVR model to predict the degree of importance for each word. For training, the input to the regression model is represented in the form <x, y>, where x is a vector of the values of features described above in this section and y is the target value. The input word for each sentence is represented as the vector x. Since we have considered 10 features, the vector length is 10 and a feature vector for a word looks like the following: <f1, f2, f3,f4, f5, f6, f7, f8, f9, f10>

For training the SVR model, each feature vector needs to be assigned a target value (y value). Since it is difficult to manually assign the degree of importance to each word, we have used an automatic process to assign the degree of importance to each word. For this purpose, we have used the DUC 2002 multi-document summarization dataset consisting of 59 folders where each folder comprises approximately 10 documents. For each folder, multiple human-created reference summaries are available in the dataset. We have used 2 reference summaries available for each input folder. The y value for each vector corresponding to a word w is calculated by the following equation,

$$yvalue = \frac{m}{2} \tag{6.12}$$

Where m is the total number of occurrences of word w in both the reference summaries. Since we have considered 2 reference summaries, we have taken the average occurrences. Here we assume that the more frequently a word is selected by the human summarizers, the more important the word is.

Thus, the final training instance for each word looks like:
<f1 , f2 , f3 ,f4 , f5 , f6 , f7 , f8 , f9 , f10, y value>

To train our SVM regression model, we treat each word in a sentence separately since a word may appear in multiple places in the document. Thus we have obtained 41519 training instances using the DUC 2002 dataset.

First We have used the DUC2004 dataset as the test dataset which is also represented in a way the training data is represented. Since our task is to generate a summary for each folder of the DUC 2004 dataset, each folder is processed separately and the feature vectors corresponding to the words occurring in the documents under the folder are submitted to the trained SVR model for predicting the word importance.

Sentence Scoring

In this section, we discuss the method for calculating sentence scores. The score of a sentence is calculated using word importance and sentence position.

Sentence scoring is done using two methods- (1) using word importance and (2) sentence position. To compute sentence score using word importance, we submit each word of a sentence (except stop words) to the trained SVR model and then the predictions (regression value) of the SVR model are summed up to assign a score to the sentence. Along with the stop words, We also exclude the less important terms from this calculation because a longer sentence contains many unimportant (noisy) words. We remove the words whose predicted scores are less than a threshold which is set to μ+σ, where μ is the mean predicted score of the words in the input and σ is the standard deviation. The following formula is used for sentence score calculation based on word importance.

$$Score(s) = \sum_{w \subset s} Predicted - Score(w) if Predicted - Score(w) >= \mu + \sigma \tag{6.13}$$

Where $Predicted - Score(w)$ is the word importance score predicted by the trained SVR model.

Since a word may occur in multiple places in a document, our feature representation method gives a different feature vector for each occurrence of a word. So, the importance score for a given word may be different in the different contexts the word appears.

Since the previous studies[77] show that the sentence position-based score is also useful in the text summarization field, we have combined the above-mentioned word importance-based sentence score with the position-based score. The positional score for the sentence S is computed using the following formula given in [114]:

$$Score^{position}(s) = max(0.5, exp\frac{-p(s^d)}{\sqrt[3]{M^d}}) \tag{6.14}$$

Where $-p(s^d)$ is the position of S in the document d, and M^d is the document size in terms of sentences.

Combining Sentence Score

The final score for a sentence is obtained by linearly combining the word importance-based score and the position-based score. The final score of each sentence S is computed as follows:

$$CombinedScore(s) = Score^{wordimportance}(s) + Score^{position}(s) \tag{6.15}$$

Where $Score^{wordimportance}(s)$ is the normalized sentence score due to word importance and $Score^{position}$ is the score assigned to the sentence due to its position in the document. For normalization, we divide $Score^{wordimportance}(s)$ by the maximum word importance-based sentence score obtained by considering all sentences in the input.

After calculating the scores of the sentences, they are ranked according to their scores obtained using equation (6.12).

Summary Generation

For summary generation, sentences are selected one by one from the ranked list. While selecting sentences in a summary, the redundant sentences are removed from the summary because the redundancy affects summary quality. A sentence is selected for a summary if its similarity with the previously selected sentences is less than a predefined threshold value.

For dealing with redundancy issues we apply the TF-IDF-based cosine similarity measure. To calculate the cosine similarity between sentences, each sentence is represented as the real-valued vector of TF*IDF weights of the words in a sentence. The following formula defines cosine similarity between two sentences S1 and S2:

$$Cosine - Similarity(S\_1, S\_2) = \frac{V(S\_1) \cdot V(S\_2)}{\mid V(S\_1) \mid \cdot \mid V(S\_2) \mid} \tag{6.16}$$

Algorithm 1 Summary Generation

```
Order the sentences in decreasing order of their final scores.
Select the top-ranked sentence first.
Select the next sentence from the ranked list if it is sufficiently dissimilar from the previously selected sentences (if the similarity of the current sentence with any of the previously selected is<= threshold value ($T_sim$)).
Continue the sentence selection one by one in this manner until the desired summary length is reached.
```

After removing the redundant sentences from the summary, we choose the most important sentences to create a summary using Algorithm 1.

6.3 Description of Dataset

We have used a Bengali multi-document summarization dataset and a benchmark English dataset for testing the proposed MDS model. For creating the Bengali test dataset, we have considered 4 Bengali queries and performed the search on the FIRE 2010 dataset for retrieving documents relevant to the queries. For each query, we consider the top 30 relevant documents and submit them to the histogram-based clustering algorithm discussed in Chapter 2. After clustering the documents retrieved in response to each query, we get a total of 31 clusters of documents. We used these 31 clusters for evaluating the proposed summarization model. For each cluster, we manually create one reference summary consisting of six sentences. For training the SVR model used to predict the Bengali word importance, we have used a dataset containing 98 document-summary pairs. These documents are collected from the Bengali Daily newspaper, Anandabazar Patrika.

To prove the generalization ability of the proposed model, we have also evaluated it using the Benchmark English multi-document summarization datasets. DUC 2002 multi-document summarization dataset is used for training the SVR which is used for predicting word importance and the proposed summarization model is tested on the DUC 2004 multi-document summarization dataset. The DUC 2002 dataset and DUC 2004 dataset contain 59 and 50 input folders respectively. For both datasets, each input folder contains approximately 10 news documents. The organizers of DUC (NIST) released both datasets along with the reference summaries. In the DUC 2002 dataset, there are two reference summaries for each folder whereas there are four reference summaries for each folder in the DUC 2004 dataset.

[0] https://www-nlpir.nist.gov/projects/duc/guidelines/2002.html

[0] https://duc.nist.gov/duc2004/

6.4 Evaluation, experiment, and Results

We have used a popular automatic summary evaluation package, called ROUGE for summary evaluation. In this case, the ROUGE version 1.5.5 [97] has been used. ROUGE package evaluates system summaries by comparing each system summary with a set of reference (model) summaries and reports evaluation scores in terms of ROUGE-N scores which are computed by counting word N-grams common between a system summary and the human summaries (reference summaries).

$$ROUGE\_N = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in N} Count_{match}(gram_n)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in N} Count(gram_n)} \tag{6.17}$$

Where n stands for the length of the n-gram, $gram_n$, and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries.

Equation 6.15 was taken from the paper authored by Lin [97]. This equation was used to compute recall. The older version of ROUGE Incorporated this equation. However, the latest ROUGE version 1.5.5 reports precision, recall, and F-score for summary evaluation. Since we have used ROUGE version 1.5.5, in this work, We have considered precision, recall, and F-score for summary evaluation. As per the DUC 2004 guidelines, the summary length is set to 665 bytes while setting the options of ROUGE. To do this, option –b 665 has been set in the ROUGE package. It means that the first 665 bytes (approximately 100 words) are taken from a system summary when it is evaluated.

6.4.1 Experiment

We have conducted several experiments to select the best model. We develop the following two models:

- Model A which uses only the word importance-based score
- Model B which uses a combined score, a linear combination of the word importance-based score, and the positional score.

Since the SVR method used for predicting word importance is highly dependent on the feature set, we initially find the optimal feature set using the backward elimination method [115]. In our work, we have applied the backward elimination method to eliminate the relatively irrelevant features. This method works as follows (Algorithm 2).

When Algorithm 2 was implemented, we used ROUGE-1 F1-Score for model evaluation.

By applying the backward elimination technique, we have determined which features should be removed to enhance system performance. We found that feature 5, that is, the feature ”TF-IDF local” is not useful because its removal reduces the sum of squared(SSE)

Algorithm 2 Backward Elimination Algorithm

```
Start with the full set of features, F.
Evaluate the model.
for i do = 1 to |F|
    Choose the feature $f_i$ from F
    Remove $f_i$ and evaluate the model with $F_i$
    If the model with $F_i$ performs better than the model with F
  features, $f_i$ is removed, otherwise, add $f_i$ to F.
    Next i
end for
```

error of the SVR model. Therefore, we exclude this feature and train the SVR model with the remaining 9 features. Another important tunable parameter for our proposed summarization model is the similarity threshold (refer to T_{sim} mentioned in Algorithm 1) used to detect and remove redundant sentences from the summary. In our experiment on both the English and Bengali datasets, Model A gives the best ROUGE-1 F1-Score when we set this threshold to 0.5. The second model, Model B, gives the best results when we set the threshold to 0.4.

6.4.2 Results

For English dataset

Table 6.1 shows Rouge-1 precision, Recall, and F-score obtained by our developed models, Model A and Model B. The ROUGE score shown in Table 6.1 indicates that the proposed second model (Model B) performs better than the first model. These results demonstrate that the positional information is also effective for multi-document summarization.

Table 6.2 shows the Rouge-2 precision, Recall, and F-score obtained by Model A and Model B. The results shown in Table 6.2 reveal that our proposed second model (Model B) outperforms the first model. From these results, we can conclude that the word importance alone is not sufficient, the positional information needs to be combined with the word importance while measuring the sentence score. The positional information is useful for the DUC 2004 dataset because this dataset is a collection of news documents and the positional feature has proven to be effective for summarizing the news documents[92]. The positional information is useful for the DUC 2004 dataset because this dataset is a collection of news documents and the positional feature has proven to be effective for summarizing the news documents[92].

For Bengali dataset

Table 6.3 shows the Rouge-1 precision, Recall, and F-score obtained by our developed models, Model A and Model B for the Bengali summarization Dataset developed by us. The ROUGE score shown in Table 6.3 indicates that the proposed second model (Model B) performs better than the first model. These results also demonstrate that the positional

Table 6.1: Rouge-1 score obtained by our proposed two models. Model A (only word importance-based) and Model B (word importance score +positional score on the English dataset)

SYSTEMS	ROUGE-1 Recall	ROUGE-1 Precision	ROUGE-1 F-score
Model B Model for English dataset	0.3837	0.3775	0.3804
Model A Model for English dataset	0.3762	0.3678	0.3717

Table 6.2: Rouge-2 score obtained by our proposed two models. Model A (only word importance-based) and Model B (word importance score + positional score on the English dataset

SYSTEMS	ROUGE-2 Recall	ROUGE-2 Precision	ROUGE-2 F-score
Model B for English dataset	0.0954	0.0942	0.0944
Model A for English dataset	0.0895	0.0852	0.0882

information is also effective for Bengali news multi-document summarization.

Table 6.4 shows the Rouge-2 precision, Recall, and F-score obtained by our developed models, Model A and Model B. The results shown in Table 6.4 reveal that our proposed second model (Model B) outperforms the first model. From these results, we can conclude that the positional information needs to be combined with the word importance for better summarization performance. The positional information is useful for the Bengali dataset because this dataset is also created from a collection of news documents.

Table 6.3: Rouge-1 score obtained by our proposed two models. Model A (only word importance-based) and Model B (word importance score + positional score on the Bengali dataset

SYSTEMS	ROUGE-1 Recall	ROUGE-1 Precision	ROUGE-1 F-score
Model B for Bengali dataset	0.6047	0.6040	0.6040
Model A for Bengali dataset	0.5905	0.5889	0.5869

Table 6.4: Rouge-2 score obtained by our proposed two models. Model A (only word importance-based) and Model B (word importance score + positional score on the Bengali dataset

SYSTEMS	ROUGE-2 Recall	ROUGE-2 Precision	ROUGE-2 F-score
Model B for Bengali dataset	0.2702	0.2692	0.2694
Model A for Bengali dataset	0.2623	0.2536	0.2521

Comparison with existing models

The DUC 2004 dataset is a benchmark dataset and many researchers used this dataset for testing their proposed MDS models. We have also compared the performance of our proposed summarization model on the DUC 2004 dataset with some MDS baseline systems. MEAD baseline [80] is one of the best MDS baselines. We have implemented MEAD with position, centroid, and length cutoff features [80]. The DUC Coverage baseline was defined by the DUC organizers when the DUC 2004 conference was held. In the DUC Coverage baseline, a summary is generated by taking the first sentence from each document of the input cluster.

The comparison results are shown in Table 6.3. As we can see from the results shown in Table 6.3, our proposed system with positional information (Model B) for multi-document text summarization performs significantly better than the MEAD baseline and the DUC coverage baseline. The proposed system that uses only word importance-based sentence score (without positional information) (Model A) also performs better than the DUC coverage baseline. there is no significant difference between the MEAD baseline and Model A. These results prove that our proposed method for predicting word importance using SVR is effective for multi-document summarization tasks.

Table 6.5: Performance comparisons of our proposed best models with some baseline models on the English Dataset

SYSTEMS	ROUGE-2 F-score	ROUGE-2 F-score
Model B	0.3804	0.0944
MEAD baseline [80]	0.3737	0.0937
Model A	0.3717	0.0882
DUC Coverage baseline	0.3451	0.0812

6.5 Chapter Summary

In this chapter, We have defined several novel features for measuring word importance (weight). Using these features, a support vector regressor is trained for predicting word weight (word importance). The sentence score is calculated by combining two types of scores -(1) sentence score which is calculated by summing up the weights (predicted by SVR) of the important words contained in the sentence, and (2) Sentence score due to the position of the sentence in the document. During summary generation, the sentences are ranked based on scores, the top-ranked sentence is selected first in the summary. The next sentence from the ranked list is selected in the summary if it is sufficiently dissimilar to the summary created so far.

To check the generalization capability of the proposed model, we tested it on two datasets, a Bengali dataset, and an English Benchmark dataset. The experimental results reveal performance improvement when the SVR model is used for predicting word weights. We also experimented with the impact of positional information on the summarization performance. We observed that the word importance alone is not sufficient. The positional information needs to be combined with the word importance while measuring the sentence score. These results agree with previous studies [92]. Our experimental results establish the effectiveness of the proposed method in multi-document text summarization for both Bengali and English datasets.

The proposed models have also been compared to two baseline multi-document summarization systems that are currently in use, the MEAD baseline system and the DUC Coverage baseline system. MEAD baseline, one of the best MDS baselines [80]. The outcomes displayed in Table 6.3 demonstrate that our suggested model, Model B, outperforms both the baselines.

7

Conclusions and Future Scope

In this chapter, we conclude the thesis. First, we summarize the previous chapters. Then we conclude and give an outline of future directions.

7.1 Summary

In this thesis, we investigate the problem of Bengali information retrieval, clustering of documents returned by an IR model, and producing a summary for each cluster. We have proposed various novel approaches for Bengali Information retrieval, and multi-document summarization that produce a summary from a cluster of related documents. Our investigations are divided as follows.

- **Cluster-based Smoothing for Document Language Model-based Bengali Information Retrieval:** We proposed in Chapter 2, a cluster-based smoothing method which is combined with a collection-based smoothing method for smoothing a document language modeling-based approach to IR. The proposed smoothing method uses cluster information where documents are clustered using a histogram-based clustering method with FastText word embedding-based document representation. Since the performance of cluster-based smoothing depends on clustering quality, we improve smoothing by enhancing the clustering algorithm. A semantic similarity function that uses FastText word embedding-based document representation is used for document clustering.

- **Semantic Methods for Bengali Information Retrieval:** Since Smoothing alone is not sufficient to deal with the word mismatch problem, we need to have a model that can do conceptual-level matching between the query and the document. In Chapter 3, we propose three semantic IR models for the Bengali language and evaluate these models using benchmark datasets. Three methods presented in this chapter are (1) the LSI-based IR model, (2) the word embedding-based IR model and (3) the BERT embedding-based IR model.

- **Hybrid Semantic Query Expansion for Bengali Information Retrieval:** Query expansion (QE) [4] is a well-known effective approach to addressing the word mismatch problem. some recent works on query expansion have shown that instead of using a single query expansion technique, the fusion of multiple expansion techniques is useful in extracting more informative query expansion terms [62, 63, 64, 65, 66].

In Chapter 4, we propose a novel hybrid query expansion framework that combines statistical, lexical, and word embedding-based semantic methods to choose the contextually and semantically related terms for query expansion leading to improving Bengali retrieval performance. The proposed approach has multiple stages. In the first stage, the initial search is performed using okapi BM25 [19]. In the second stage, the candidate expansion terms are extracted using three methods (1)a lexical method that uses a Bengali WordNet, (2) a word embedding-based method that uses the top $k_{d}^{(}(emb))$ documents from the initial search results and applies the word-to-word semantic similarity measure to extract the words which are highly similar to the query words, and (3) a term frequency-based method that uses term frequency statistics for extracting words from the initial search results. At the third stage of the proposed approach, the candidate expansion terms extracted using three different extraction methods are combined to create a pool of candidate expansion terms which are ranked based on their scores where the score of a candidate term is a linear combination of its contextual score and frequency score. After ranking the candidate terms, a certain number of top-ranked terms are selected as the final expansion terms and they are added to the original query, and the search is again performed using Bengali Information Retrieval Using a Model Combination of the expanded query.

- **Bengali Information Retrieval Using Model Combination:**

  Every IR model has a set of assumptions, and every model has multiple scenarios in which it fails. There might be another IR model that performs better on certain queries when the first model yields poor results. The reason for this is that the relative significance of the query words found in the documents is not accounted for by the semantic model. Since the semantic-based approach reduces precision while increasing recall, it is insufficient on its own. In Chapter 5, we propose two hybrid IR models that are developed using model combination, (1) Hybrid model-1 which combines a TFIDF-based vector space model(VSM) with the latent semantic indexing(LSI) model, and (2) Hybrid model-2 which combines a TFIDF-based vector space model (VSM) with the word embedding-based model.

- **Clustering and Summarizing Search Results:** When a user submits a query to the search engine, thousands of text documents are returned by search engines. This brings on the issue of information overload. In Chapter 6, we propose a method that can group the search results returned by an IR model into multiple clusters using a clustering algorithm and summarize each cluster of documents using a multi-document text summarization system. For clustering search results, we have used the histogram-based clustering algorithm discussed in Chapter 2. For summarizing clusters of related documents, We propose an extractive multi-document summarization method that assigns weights to the words using a support vector regressor trained using a predefined set of novel features designed by us. Multi-document summarization(MDS) system produces a single summary for each cluster of related documents[5].

The main idea behind clustering and summarizing search results is to display summaries of the clusters first to the user and the user will read the summary to decide which cluster may contain the most relevant documents. When the most relevant cluster is clicked by the users, the documents belonging to the cluster are displayed to the users. Thus it reduces the information search time.

7.2 Contribution of the Thesis

The main contributions of the thesis have been discussed in the concerned chapters, in this section, we summarize the current thesis's contribution in Table 7.1.

Table 7.1: Contributions summary of the Thesis

Chapter	Problem	Contribution
Chapter 2	Can a better smoothing method improve the Bengali IR performance?	We explored various smoothing techniques for improving the performance of Bengali information retrieval. To improve the language model for Bengali information retrieval, we have explored a hybrid smoothing method that combines the cluster-based smoothing method with the collection-based smoothing method. with the document language model

Continued on next page

Table 7.1 – continued from previous Table

Chapter	Problem	Contribution
Chapter 3	To what extent, can the semantic method solve the word mismatch problem? Which is the suitable semantic method for Bengali iR?	We have explored the LSI-based method, the Word embedding-based method, and the BERT-based method for Bengali IR. We have also explored several combinations of these three semantic methods for finding a suitable hybrid method for Bengali IR.
Chapter 4	Can the query expansion (QE) method improve the Bengali IR performance?	We have explored a hybrid query expansion method that uses statistical, lexical, and word embedding-based methods for selecting suitable expansion terms that are added to the query, and then the search is performed with the expanded query
Chapter 5	Can IR model combination improve the Bengali information retrieval performance?	We have explored several combinations of the semantic and lexical IR models for developing a hybrid model that can improve Bengali information retrieval.
Chapter 6	How can the search results be presented to the users so that they can quickly find out the most relevant information?	We have explored the clustering of documents returned by an IR model and the multi-document summarization method that produces a single summary for each of the clusters produced by the histogram-based clustering algorithm.

7.3 Limitation of the Thesis

Although we have discussed the limitations of the proposed methods in the concerned chapters, in this section, we summarize the current thesis flaws:

- As mentioned in Chapter 2, the proposed cluster-based smoothing method has some limitations. It does not perform equally well for all types of queries. To investigate the reasons, we manually verified the query-document combination and discovered that it was less effective when the query was too short and so, the document contained no query words. This is why the proposed smoothing technique was not effective for the short as well as complex abstract query.

- The semantic methods in Chapter 3 are used to deal with the situation when the query is highly abstract. However, the proposed semantic models individually cannot achieve the baseline results. The possible reasons that we identify are the semantic model shows relatively high similarity values when the queries and documents are compared. Therefore, the semantic model achieves high recall but low precision. For example, the Word embedding-based model does not take into account the relative importance of terms in documents and queries although it improves recall. To deal with this issue, we use a two-stage approach: in the first stage, we use BM25 for producing the initial search results and then we use BERT-based methods for re-ranking the documents. Although this two-stage approach improves overall performance, due to computational resource constraints, we have used the BERTbase model.

- In Chapter 4, we introduced a hybrid query expansion method that extracts expansion terms from the top *k* documents chosen from the initial search results returned by the original query. It also uses synonyms of the query words retrieved from WordNet. Since the synonym-based query expansion alone is not sufficient, the proposed query expansion method could not perform well when the top k retrieved documents were less relevant to the original query.

- Chapter 5 introduced a model that combines two IR systems. The main limitation of this combined model is that it is slower than the individual IR model.

- Chapter 6 introduced a supervised term weighting method that uses an SVR (Support Vector Regressor) which is trained on a set of features. When this term weighting method is incorporated into the multi-document summarization system, the summarization performance is improved. However, the proposed term weighting is not free from flaws. If it could be possible to include more effective features in the feature set, the term weighting might be more accurate. The redundancy removal method used for reducing redundancy in the summary also failed in some cases to remove redundant information.

7.4 Conclusion

Finding an effective approach to alleviating the term mismatch problem for improving the performance of Bengali Information retrieval is the prime objective of the thesis. We also investigated finding a method for clustering documents retrieved by an IR model and producing summaries from the clusters of documents to enable the users to quickly satisfy their information needs.

We have explored various lexical, probabilistic, semantic, query expansion, and hybrid approaches to information retrieval. These approaches include

- TF-IDF-based approach
- Okapi BM25 approach
- Language modeling approach with clustering-based smoothing
- Word embedding-based and BERT embedding approaches
- Hybrid query expansion approach that combines statistical, lexical, and word embedding-based semantic methods
- Model combination approach

Since an IR system creates a new issue by returning an excessive number of documents for a single query. To determine which documents are more useful, the user frequently has to search through a huge number of documents to discover a few that meet their information needs. This is likewise a time-consuming task. Clustering search results and summarizing each cluster enables the users to find the relevant information quickly. To deal with this information overload problem, we also focus on summarising a cluster of related documents where the clusters are produced by applying a document clustering algorithm on the documents retrieved by an IR system.

All the above-mentioned IR approaches and the multi-document summarization approach have been tested on the benchmark datasets. We have used the Mean Average Precision(MAP), a well-known IR evaluation metric for evaluating the proposed IR models, and the popular automatic evaluation package called ROUGE(Recall-Oriented Understudy for Gisting Evaluation) summary evaluation metrics. Finally, we conclude with the following observations:

- In Chapter 2, we present an improved cluster-based smoothing method integrated with a language modeling approach to Bengali information retrieval for improving document retrieval performance. The experimental results reveal that retrieval performance can be improved when a better smoothing method is used. Since cluster-based smoothing heavily depends on the quality of clusters produced, we observe that the incorporation of the semantic similarity measure in the clustering process affects the smoothing effect. To do this, we represent each document using word embedding. Our experimental results establish the fact that, for word embeddings-based document presentation, using vectors of selected keywords is more useful than using those of all words. The best model proposed in Chapter 2 performs significantly better than the well-known IR model BM25. We think that the more deep semantic analysis is needed to solve very complex query situations. To tackle this, the smoothing alone is not sufficient, the model should be able to do conceptual-level matching between the query and the document. The semantic approaches can be effective in this situation.

- In Chapter 3, we present mainly three types of semantic IR models: the LSI-based IR model, the static word embedding-based semantic IR model, and the BERT-based semantic IR model. The experimental results suggest that the IR model that uses BM25 for generating initial search results and re-ranks initial results based on a combination of BM25 and BERT scores performs the best among all semantic IR models presented in this chapter. The proposed model also outperforms a recently published IR model[45] that has used BERT-based semantic representation.

  We observe that The best semantic IR model should be a two-step ranking-based IR model where, in the first step, documents are ranked using the traditional IR model like BM25, and in the second step, the documents returned at the first step are re-ranked using BM25 score + BERT-score. The reason for this combination is that the BERT score itself is not enough to re-rank the documents because it does not capture the relative importance of query words present in the documents.

- In Chapter 4, We have presented a hybrid query expansion approach that combines the lexical, semantic, and statistical methods for extracting expansion terms. To improve the Bengali information retrieval performance, the query is expanded with the extracted expansion terms. WordNet is used in the lexical technique to extract potential expansion terms that are synonymous with the query terms; in contrast, the statistical and semantic methods are integrated into the pseudo-relevance feedback framework to extract candidate expansion terms. While the semantic approach extracts terms based on word embedding-based similarity between the original query terms and the terms occurring in the top-ranked documents selected from the initial search results, the statistical method uses term frequency statistics to extract candidate terms from some top-ranked documents selected from the initial search results. The candidate expansion terms extracted by these three methods are combined to create a pool that is further processed to choose the final expansion terms which are contextually similar to the query terms.

  We observe that the hybrid query expansion method is more effective than an individual query expansion method for Bengali information retrieval. The synonym-based query expansion is not alone effective for Bengali information, but it is useful when it is combined with other query expansion methods. The experimental results reveal that the hybrid approach that combines WordNet-based, Word embedding-based, and frequency-based candidate term extraction methods is superior to other possible hybrid models developed by us.

- In Chapter 5, we present a hybrid IR model that combines the relevance scores produced by two different individual IR models for each document-query pair and ranks documents based on the combined scores. Our experiments show that our proposed hybrid model performs better than the individual component model. One important factor for the IR task is to make the retrieval process faster. When the model combination is made, the hybrid system becomes relatively slow.

- In Chapter 6, we present a multi-document summarization method that uses a supervised term weighting method. The proposed term weighting method uses the support vector regression model for predicting word importance. We observe that, when the SVR model is used for predicting word importance, the summarization performance is improved. We also experimented with the impact of positional information on summarization performance. We observe that sentence position information is effective in multi-document text summarization. We also observe that clustering search results and displaying cluster summaries to the users enables them to find relevant documents quickly and efficiently.

7.4.1 Scope of the Future Research

Although we have made every effort to ensure that this thesis is as comprehensive as possible, there is still room for improvement. Some such future directions are summarized in this section.

- In this thesis, we have explored information retrieval for one Indian language, Bengali. In the future, we would like to work on information retrieval for other low-resource Indian languages like Hindi, Marathi, etc.

- The performance of our proposed clustering-based smoothing model described in Chapter 2 can further be enhanced by using a more powerful document representation method because clustering performance is heavily dependent on document representation. For better document representation, the most recent language models can be used.

- In Chapter 3, the best semantic IR model is a two-step ranking model. In the first step, the BM25 model is used for producing the initial results and then, in the second step, the BERT model is used for document representation and re-ranking. This IR model can be improved by improving the re-ranking step.

- In Chapter 4, we have proposed a hybrid query expansion method that combines three candidate term extraction techniques. In future work, we will look for better candidate term extraction and final term selection strategies for the query expansion model.

- In Chapter 5, we presented an IR model that combines two IR models. In the future, we would like to combine more than two IR models. Since model combination slows down the retrieval performance, we need to look for novel indexing strategies so that model combination does not take much time while retrieving documents.

- In Chapter 6, clustering of the documents retrieved by an IR model is done and each cluster of documents is summarized for displaying cluster summaries to the users so that the users can quickly identify which cluster may contain the most relevant documents. For clustering documents, we have used a histogram-based clustering method. In the future, we would like to search for an efficient document clustering algorithm for organizing the search results in a better way. For summarizing a cluster of documents, we have used a multi-document summarization method that weighs terms using a support vector regressor trained with a set of hand-crafted features. In the future, we would like to generate an abstract summary for each cluster. The deep learning-based sentence fusion model can be used for this purpose.

Bibliography

[1] S. Chatterjee and K. Sarkar, “Combining ir models for bengali information retrieval,” *International Journal of Information Retrieval Research (IJIRR)*, vol. 8, no. 3, pp. 68–83, 2018.

[2] A. Das, B. Kundu, L. Ghorai, A. K. Gupta, and S. Chakraborti, “Anwesha: A tool for semantic search in bangla,” 2021.

[3] D. Radev and W. Fan, “Automatic summarization of search engine hit lists,” in *ACL-2000 Workshop on Recent Advances in Natural Language Processing and Information Retrieval*, 2000, pp. 99–109.

[4] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval.* Cambridge university press, 2008.

[5] D. Radev, E. Hovy, and K. McKeown, “Introduction to the special issue on summarization,” *Computational linguistics*, vol. 28, no. 4, pp. 399–408, 2002.

[6] P. Majumder, M. Mitra, S. K. Parui, G. Kole, P. Mitra, and K. Datta, “Yass: Yet another suffix stripper,” *ACM transactions on information systems (TOIS)*, vol. 25, no. 4, pp. 18–es, 2007.

[7] J. H. Paik and S. K. Parui, “A simple stemmer for inflectional languages,” in *Forum for Information Retrieval Evaluation*. Citeseer, 2008.

[8] K. Sarkar and A. Gupta, “An empirical study of some selected ir models for bengali monolingual information retrieval,” *arXiv preprint arXiv:1706.03266*, 2017.

[9] T. D. Roy, S. Khatun, R. Begum *et al.*, “Vector space model based topic retrieval from bengali documents,” in *2018 International Conference on Innovations in Science, Engineering and Technology (ICISET)*. IEEE, 2018, pp. 60–63.

[10] G. Salton, A. Wong, and C.-S. Yang, “A vector space model for automatic indexing,” *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[11] P. Bhaskar, A. Das, P. Pakray, and S. Bandyopadhyay, “Theme based english and bengali ad-hoc monolingual information retrieval in fire 2010,” *Corpus*, vol. 1, pp. 25–586, 2010.

[12] S. Das and P. Mitra, “A rule-based approach of stemming for inflectional and derivational words in bengali,” in *IEEE Technology Students' Symposium*. IEEE, 2011, pp. 134–136.

[13] M. Kowsher, I. Hossen, and S. Ahmed, “Bengali information retrieval system (birs),” *International Journal on Natural Language Computing (IJNLC)*, vol. 8, no. 5, 2019.

[14] S. Bandhyopadhyay, A. Das, and P. Bhaskar, "English bengali ad-hoc monolingual information retrieval task result at fire 2008," in *Working Note of Forum for FIRE-2008*, 2008.

[15] U. Barman, P. Lohar, P. Bhaskar, and S. Bandyopadhyay, "Ad-hoc information retrieval focused on wikipedia based query expansion and entropy based ranking," *corpus*, vol. 4, pp. 57–370, 2012.

[16] S. E. Robertson, "The probability ranking principle in ir," *Journal of documentation*, 1977.

[17] S. E. Robertson and K. S. Jones, "Relevance weighting of search terms," *Journal of the American Society for Information science*, vol. 27, no. 3, pp. 129–146, 1976.

[18] S. Robertson, H. Zaragoza *et al.*, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[19] K. S. Jones, S. Walker, and S. E. Robertson, "A probabilistic model of information retrieval: development and comparative experiments: Part 2," *Information processing & management*, vol. 36, no. 6, pp. 809–840, 2000.

[20] J. M. Ponte and W. B. Croft, "A language modeling approach to information retrieval," in *ACM SIGIR Forum*, vol. 51, no. 2. ACM New York, NY, USA, 2017, pp. 202–208.

[21] H. Fang, T. Tao, and C. Zhai, "A formal study of information retrieval heuristics," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004, pp. 49–56.

[22] D. Gungaly and M. Mitra, "Using language modeling at fire 2008 bengali monolingual track," in *Working Notes of the Forum for Information Retrieval Evaluation (FIRE'08)*. Citeseer, 2008.

[23] L. Dolamic and J. Savoy, "Unine at fire 2008: Hindi, bengali, and marathi ir," in *Working Notes of the Forum for Information Retrieval Evaluation*, 2008, pp. 12–14.

[24] D. Harman, "How effective is suffixing? jotrnal of the american society for information," *Science*, vol. 42, no. 1, pp. 321–331, 1991.

[25] J. H. Paik, M. Mitra, S. K. Parui, and K. Järvelin, "Gras: An effective and efficient stemming algorithm for information retrieval," *ACM Transactions on Information Systems (TOIS)*, vol. 29, no. 4, pp. 1–24, 2011.

[26] G. Amati and C. J. Van Rijsbergen, “Probabilistic models of information retrieval based on measuring the divergence from randomness,” *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 357–389, 2002.

[27] S. Gupta and D. Garg, “Comparison of semantic and syntactic information retrieval system on the basis of precision and recall,” *Int J Data Eng*, vol. 2, no. 3, pp. 93–101, 2011.

[28] A. Singhal and F. Pereira, “Document expansion for speech retrieval,” in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999, pp. 34–41.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *Advances in neural information processing systems*, vol. 26, 2013.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

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

[33] S. Chatterjee and K. Sarkar, “Bengali document retrieval using model combination,” *3rd International conference on Frontiers in Computing and Systems (COMSYS-2022)*, vol. 8, no. 3, pp. 68–83, 2018.

[34] M. W. Berry, S. T. Dumais, and G. W. O’Brien, “Using linear algebra for intelligent information retrieval,” *SIAM review*, vol. 37, no. 4, pp. 573–595, 1995.

[35] M. N. Hoque, R. Islam, and M. S. Karim, “Information retrieval system in bangla document ranking using latent semantic indexing,” in *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. IEEE, 2019, pp. 1–5.

[36] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[37] D. Ganguly, J. Leveling, and G. J. Jones, “A case study in decompounding for bengali information retrieval,” in *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 2013, pp. 108–119.

[38] S. Kuzi, A. Shtok, and O. Kurland, “Query expansion using word embeddings,” in *Proceedings of the 25th ACM international on conference on information and knowledge management*, 2016, pp. 1929–1932.

[39] D. Roy, D. Paul, M. Mitra, and U. Garain, “Using word embeddings for automatic query expansion,” *arXiv preprint arXiv:1606.07608*, 2016.

[40] G. Zuccon, B. Koopman, P. Bruza, and L. Azzopardi, “Integrating and evaluating neural word embeddings in information retrieval,” in *Proceedings of the 20th Australasian document computing symposium*, 2015, pp. 1–8.

[41] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, “Learning word vectors for 157 languages,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. [Online]. Available: https://aclanthology.org/L18-1550

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[43] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” *Advances in neural information processing systems*, vol. 32, 2019.

[44] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: A lite bert for self-supervised learning of language representations,” *arXiv preprint arXiv:1909.11942*, 2019.

[45] A. Das, J. Acharya, B. Kundu, and S. Chakraborti, “Revisiting anwesha: Enhancing personalised and natural search in bangla,” in *Proceedings of the 19th International Conference on Natural Language Processing (ICON)*, 2022, pp. 183–193.

[46] H. K. Azad and A. Deepak, “Query expansion techniques for information retrieval: a survey,” *Information Processing & Management*, vol. 56, no. 5, pp. 1698–1735, 2019.

[47] C. J. Crouch and B. Yang, “Experiments in automatic statistical thesaurus construction,” in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, 1992, pp. 77–88.

[48] A. Natsev, A. Haubold, J. Tešić, L. Xie, and R. Yan, “Semantic concept-based query expansion and re-ranking for multimedia retrieval,” in *Proceedings of the 15th ACM international conference on Multimedia*, 2007, pp. 991–1000.

[49] B. Al-Shboul and S.-H. Myaeng, “Query phrase expansion using wikipedia in patent class search,” in *Asia Information Retrieval Symposium*. Springer, 2011, pp. 115–126.

[50] S. Ganesh and V. Varma, “Exploiting structure and content of wikipedia for query expansion in the context,” in *Proceedings of the International Conference RANLP-2009*, 2009, pp. 103–106.

[51] E. M. Voorhees, “The trec robust retrieval track,” in *ACM SIGIR Forum*, vol. 39, no. 1. ACM New York, NY, USA, 2005, pp. 11–20.

[52] Y. Xu, G. J. Jones, and B. Wang, “Query dependent pseudo-relevance feedback based on wikipedia,” in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 59–66.

[53] J. Rocchio, “Relevance feedback in information retrieval,” *The Smart retrieval system-experiments in automatic document processing*, pp. 313–323, 1971.

[54] G. Salton and C. Buckley, “Improving retrieval performance by relevance feedback,” *Journal of the American society for information science*, vol. 41, no. 4, pp. 288–297, 1990.

[55] C. Buckeley, G. Salton, J. Allan, and A. Stinghal, “Automatic query expansion using smart,” in *Proceedings of the 3rd Text Retrieval Conference*, 1994, pp. 69–80.

[56] M. Mitra, A. Singhal, and C. Buckley, “Improving automatic query expansion,” in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998, pp. 206–214.

[57] A. Atreya, A. Kankaria, P. Bhattacharyya, and G. Ramakrishnan, “Query expansion in resource-scarce languages: A multilingual framework utilizing document structure.” *ACM Trans. Asian Low Resour. Lang. Inf. Process.*, vol. 16, no. 2, pp. 14–1, 2016.

[58] R. R. Prasath and S. Sarkar, “Query expansion using prf-cbd approach for documents retrieval,” in *International Conference on Pattern Recognition and Machine Intelligence*. Springer, 2013, pp. 495–500.

[59] A. L. Kaczmarek, “Interactive query expansion with the use of clustering-by-directions algorithm,” *IEEE Transactions on Industrial Electronics*, vol. 58, no. 8, pp. 3168–3173, 2010.

[60] D. Ganguly, J. Leveling, and G. J. Jones, “Exploring sentence level query expansion in language modeling based information retrieval,” 2010.

[61] M. ALMasri, C. Berrut, and J.-P. Chevallet, “A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information,” in *European conference on information retrieval*. Springer, 2016, pp. 709–715.

[62] H. ALMarwi, M. Ghurab, and I. Al-Baltah, “A hybrid semantic query expansion approach for arabic information retrieval,” *Journal of Big Data*, vol. 7, no. 1, pp. 1–19, 2020.

[63] L. Han and G. Chen, “Hqe: A hybrid method for query expansion,” *Expert Systems with Applications*, vol. 36, no. 4, pp. 7985–7991, 2009.

[64] D. K. Sharma, R. Pamula, and D. S. Chauhan, “Query expansion–hybrid framework using fuzzy logic and prf,” *Measurement*, vol. 198, p. 111300, 2022.

[65] Z. Wang and N. Qiang, “Research on hybrid query expansion algorithm,” *International Journal of Hybrid Information Technology*, vol. 5, no. 2, pp. 207–212, 2012.

[66] M. A. Zingla, C. Latiri, P. Mulhem, C. Berrut, and Y. Slimani, “Hybrid query expansion model for text and microblog information retrieval,” *Information Retrieval Journal*, vol. 21, no. 4, pp. 337–367, 2018.

[67] K. Sarkar, “A keyphrase-based approach to text summarization for english and bengali documents,” *International Journal of Technology Diffusion (IJTD)*, vol. 5, no. 2, pp. 28–38, 2014.

[68] K. Sarkar and S. Dam, “Exploiting semantic term relations in text summarization,” *International Journal of Information Retrieval Research (IJIRR)*, vol. 12, no. 1, pp. 1–18, 2022.

[69] Y. Sankarasubramaniam, K. Ramanathan, and S. Ghosh, “Text summarization using wikipedia,” *Information Processing & Management*, vol. 50, no. 3, pp. 443–461, 2014.

[70] G. Erkan and D. R. Radev, “Lexrank: Graph-based lexical centrality as salience in text summarization,” *Journal of artificial intelligence research*, vol. 22, pp. 457–479, 2004.

[71] L. Suanmali, N. Salim, and M. S. Binwahlan, “Fuzzy logic based method for improving text summarization,” *arXiv preprint arXiv:0906.4690*, 2009.

[72] M. G. Ozsoy, F. N. Alpaslan, and I. Cicekli, “Text summarization using latent semantic analysis,” *Journal of Information Science*, vol. 37, no. 4, pp. 405–417, 2011.

[73] I. V. Mashechkin, M. Petrovskiy, D. Popov, and D. V. Tsarev, “Automatic text summarization using latent semantic analysis,” *Programming and Computer Software*, vol. 37, no. 6, pp. 299–305, 2011.

[74] N. Rane and S. Govilkar, "Recent trends in deep learning based abstractive text summarization," *Int. J. Recent Technol. Eng.*, vol. 8, no. 3, pp. 3108–3115, 2019.

[75] O. Klymenko, D. Braun, and F. Matthes, "Automatic text summarization: A state-of-the-art review." *ICEIS (1)*, pp. 648–655, 2020.

[76] H. Lin and V. Ng, "Abstractive summarization: A survey of the state of the art," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 9815–9822.

[77] K. Sarkar, "Using domain knowledge for text summarization in medical domain," *International Journal of Recent Trends in Engineering*, vol. 1, no. 1, p. 200, 2009.

[78] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of research and development*, vol. 2, no. 2, pp. 159–165, 1958.

[79] K. Sarkar, "Automatic keyphrase extraction from bengali documents: A preliminary study," in *2011 Second International Conference on Emerging Applications of Information Technology*. IEEE, 2011, pp. 125–128.

[80] D. R. Radev, H. Jing, M. Styś, and D. Tam, "Centroid-based summarization of multiple documents," *Information Processing & Management*, vol. 40, no. 6, pp. 919–938, 2004.

[81] C. Shen and T. Li, "Multi-document summarization via the minimum dominating set," in *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, 2010, pp. 984–992.

[82] T. Berg-Kirkpatrick, D. Gillick, and D. Klein, "Jointly learning to extract and compress," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 481–490.

[83] W.-t. Yih, J. Goodman, L. Vanderwende, and H. Suzuki, "Multi-document summarization by maximizing informative content-words." in *IJCAI*, vol. 7, 2007, pp. 1776–1782.

[84] H. Takamura and M. Okumura, "Text summarization model based on maximum coverage problem and its variant," in *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, 2009, pp. 781–789.

[85] R. Sipos, P. Shivaswamy, and T. Joachims, "Large-margin learning of submodular summarization models," in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 2012, pp. 224–233.

[86] J. R. Thomas, S. K. Bharti, and K. S. Babu, “Automatic keyword extraction for text summarization in e-newspapers,” in *Proceedings of the international conference on informatics and analytics*, 2016, pp. 1–8.

[87] V. Ravinuthala and S. R. Chinnam, “A keyword extraction approach for single document extractive summarization based on topic centrality,” *Int. J. Intell. Eng. Syst*, vol. 10, no. 5, pp. 153–161, 2017.

[88] K. Sarkar, “Automatic single document text summarization using key concepts in documents,” *Journal of information processing systems*, vol. 9, no. 4, pp. 602–620, 2013.

[89] J. Kupiec, J. Pedersen, and F. Chen, “A trainable document summarizer,” in *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, 1995, pp. 68–73.

[90] A. Celikyilmaz and D. Hakkani-Tur, “A hybrid hierarchical model for multi-document summarization,” in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010, pp. 815–824.

[91] M. Litvak, M. Last, and M. Friedman, “A new approach to improving multilingual summarization using a genetic algorithm,” in *Proceedings of the 48th annual meeting of the association for computational linguistics*, 2010, pp. 927–936.

[92] K. Hong and A. Nenkova, “Improving the estimation of word importance for news multi-document summarization,” in *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, 2014, pp. 712–721.

[93] Y. Bidulya and A. Spryiskov, “The summarization of search results,” in *2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT)*, 2017, pp. 1–3.

[94] C. Pasupathi, B. Ramachandran, and S. Karunakaran, “Selection based comparative summarization of search results using concept based segmentation,” in *International Conference on Web and Semantic Technology*. Springer, 2011, pp. 655–664.

[95] Z. Li, J. Tang, X. Wang, J. Liu, and H. Lu, “Multimedia news summarization in search,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 7, no. 3, pp. 1–20, 2016.

[96] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008, vol. 39.

[97] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries,” in *Text summarization branches out*, 2004, pp. 74–81.

[98] C. Van Rijsbergen, “Information retrieval: theory and practice,” in *Proceedings of the Joint IBM/University of Newcastle upon Tyne Seminar on Data Base Systems*, vol. 79, 1979.

[99] H. Turtle and W. B. Croft, “Evaluation of an inference network-based retrieval model,” *ACM Transactions on Information Systems (TOIS)*, vol. 9, no. 3, pp. 187–222, 1991.

[100] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, “Learning word vectors for 157 languages,” *arXiv preprint arXiv:1802.06893*, 2018.

[101] X. Liu and W. B. Croft, “Cluster-based retrieval using language models,” in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004, pp. 186–193.

[102] K. M. Hammouda and M. S. Kamel, “Efficient phrase-based document indexing for web document clustering,” *IEEE Transactions on knowledge and data engineering*, vol. 16, no. 10, pp. 1279–1296, 2004.

[103] J. Bhogal, A. MacFarlane, and P. Smith, “A review of ontology based query expansion,” *Information processing & management*, vol. 43, no. 4, pp. 866–886, 2007.

[104] S. R. Chowdhury, K. Sarkar, and S. Dam, “An approach to generic bengali text summarization using latent semantic analysis,” in *2017 International Conference on Information Technology (ICIT)*. IEEE, 2017, pp. 11–16.

[105] S. Sarker, “Banglabert: Bengali mask language model for bengali language understading,” 2020. [Online]. Available: https://github.com/sagorbrur/bangla-bert

[106] Z. A. Yilmaz, S. Wang, W. Yang, H. Zhang, and J. Lin, “Applying bert to document retrieval with birch,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, 2019, pp. 19–24.

[107] M. Grbovic, N. Djuric, V. Radosavljevic, F. Silvestri, and N. Bhamidipati, “Context- and content-aware embeddings for query rewriting in sponsored search,” in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 383–392.

[108] G. Zheng and J. Callan, “Learning to reweight terms with distributed representations,” in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 575–584.

[109] Z. Ye and J. X. Huang, “A simple term frequency transformation model for effective pseudo relevance feedback,” in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 2014, pp. 323–332.

[110] A. B. Goldberg, D. Andrzejewski, J. Van Gael, B. Settles, X. Zhu, and M. Craven, “Ranking biomedical passages for relevance and diversity: University of wisconsin, madison at trec genomics 2006.” in *TREC*, 2006.

[111] A. Divoli, M. A. Hearst, P. Nakov, A. S. Schwartz, and A. Ksikes, “Biotext team report for the trec 2006 genomics track.” in *TREC*, 2006.

[112] K. Sarkar, “Sentence clustering-based summarization of multiple text documents,” *TECHNIA–International Journal of Computing Science and Communication Technologies*, vol. 2, no. 1, pp. 325–335, 2009.

[113] K. Sarkar, M. Nasipuri, and S. Ghose, “Using machine learning for medical document summarization,” *International Journal of Database Theory and Application*, vol. 4, no. 1, pp. 31–48, 2011.

[114] S. Lamsiyah, A. El Mahdaouy, B. Espinasse, and S. E. A. Ouatik, “An unsupervised method for extractive multi-document summarization based on centroid approach and sentence embeddings,” *Expert Systems with Applications*, vol. 167, p. 114152, 2021.

[115] K. Sarkar and S. K. Shaw, “A memory-based learning approach for named entity recognition in hindi,” *Journal of Intelligent Systems*, vol. 26, no. 2, pp. 301–321, 2017.