Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Software Department

Extracting Key-phrase Embedding using deep Average Network and Maximal Marginal Relevance To enhance Information Retrieval

A Thesis

Submitted to the Council of the College of Information Technology, the University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology, Software

By:

Alyaa Abdual Kadhum Hadi

Supervised by:

Prof. Dr. Wafaa Mohammed Saeed

2023AD ***1444 AH***

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

إِنَّا فَتَحْنَا لَكَ فَتْحًا مُّبِينًا﴿١﴾

لِيَغْفِرَ لَكَ اللَّهُ مَاتَقَدَّمَ مِن ذَنبِكَ وَمَا تَأَخَّرَ وَيُتِمَّ

نِعْمَتَهُ عَلَيْكَ وَيَهْدِيَكَ صِرَاطًا مُّسْتَقِيمًا﴿٢﴾

وَيَنصُرَكَ اللَّهُ نَصْرًا عَزِيزًا﴿٣﴾

سورة الفتح

الآية ١-٣

Supervisor Certification

I certify that the thesis entitled (**Extracting Key-phrase Embedding using deep Average Network and Maximal Marginal Relevance To enhance Information Retrieval**) was prepared under my supervision at the Department of Software/ College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Master in Information Technology,Software.

Signature:

Supervisor Name: Prof. Dr. Wafaa Mohammed Saeed
Date: / / 2023

The Head of the Department Certification

Given the available recommendations, I forward the thesis entitled "**Extracting Key-phrase Embedding using deep Average Network and Maximal Marginal Relevance To enhance Information Retrieval**" for debate by the examination committee.

Signature:

Prof. Dr . Ahmed Saleem Abbas

Head of Software Department

Date: / / 2023

Declaration

I hereby declare that this thesis, submitted to the University of Babylon in partial fulfillment of the requirement for the degree of Master in Information Technology, Software, has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for experts and summaries whose source is appropriately cited in the references.

Signature:

Name:Alyaa Abdual Kadhum Hadi

Date: / / 2023

Dedication

My thanks and gratitude to my family deserves thanks for being the main pillar and my source of strength in life. My dear father spent his life in order to reach our ambitions, and his words were like the light that illuminates my path and overcome obstacles. My mother did not miss a corner of prayer without praying for me.

My thanks to my dear sister was a supporter of my step by encouraging me. My siblings, friends, grateful to all of you.

My thanks to everyone who encouraged me to continue and achieve my dream, even with a word! In the end, I owe it to everyone who believed in my ability to realize my ambitions. I am indebted to my family for their constant giving, love, and faith in me.

Sincerely.

Alyaa Abdual Kadhum Hadi

Certification of the Examination Committee

We, the undersigned, certify that (Alyaa Abdual Kadhum Hadi), candidate for the degree of Master in Information Technology, Software, has presented her thesis of the following title (**Extracting Key-phrase Embedding using deep Average Network and Maximal Marginal Relevance To enhance Information Retrieval)** as it appears on the title page and front cover of the thesis, that the said thesis is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: 21/6/2023.

Signature:
Name: Prof. Dr. Huda Naji Nawaf
Date: / / 2023
(Chairman)

Signature:	Signature:
Name: Prof. Dr. Ayad Rodhan Abbas	Name: Asst. Prof. Dr. Safa Saad Abbas
Date: / / 2023	Date: / / 2023
(Member)	**(Member)**

Signature:
Name: Prof. Dr. Wafaa Mohammed Saeed
Date: / / 2023
(Supervisor)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: Prof. Dr. Hussein Atiya Lafta
Date: / / 2023
(Dean of the College of Information Technology)

Acknowledgment

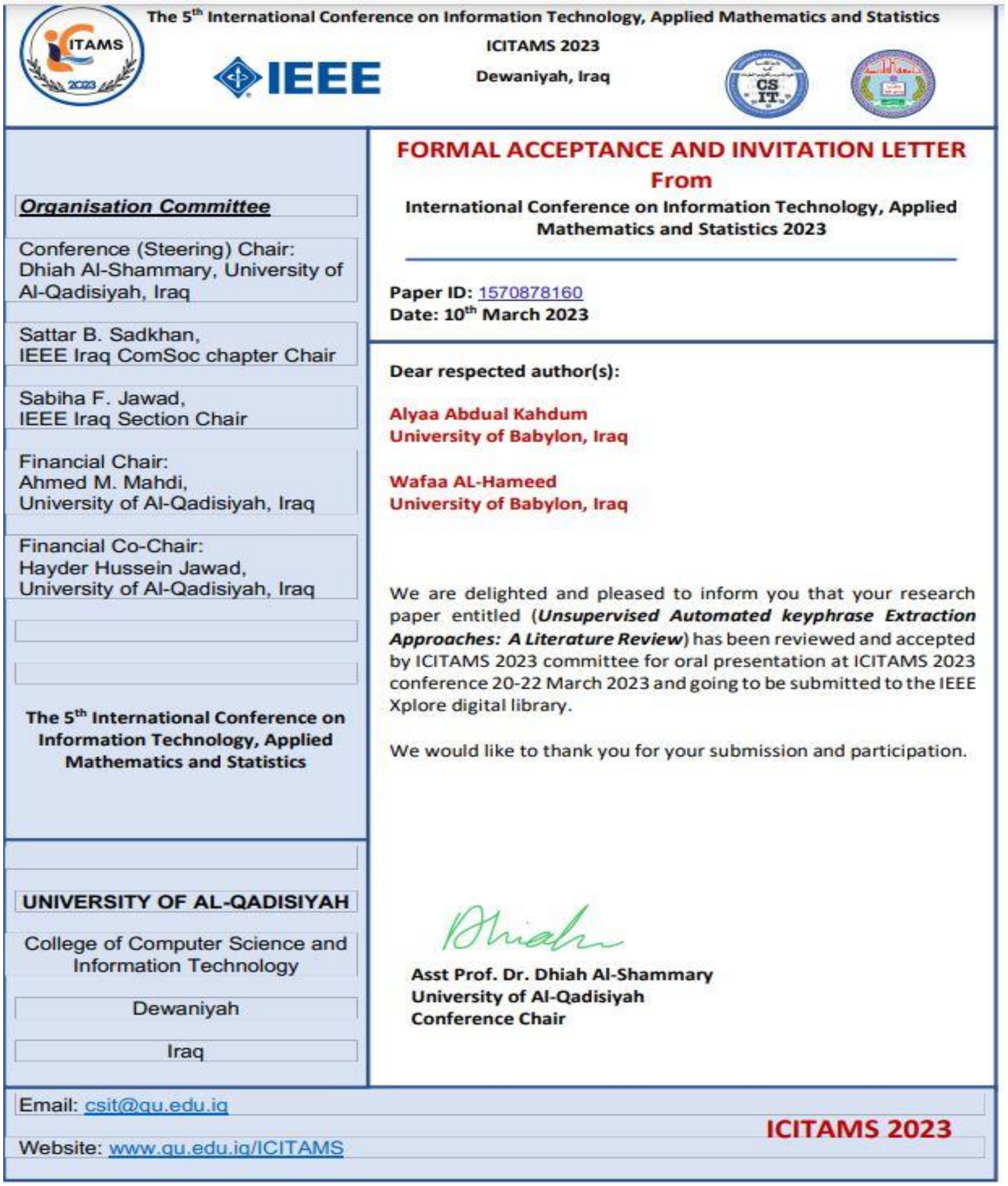
First and foremost, praises and thanks to God, the Almighty, for His showers of blessings throughout my work to complete the research successfully.

I would like to express my deep and sincere gratitude to my thesis supervisor, Dr. Wafaa Mohammed Saeed for allowing me to do research and providing invaluable guidance throughout this research. Her dynamism, vision, sincerity, and motivation have deeply inspired me. she has taught me the methodology to carry out the research and to present the research works as clearly as possible. Working and studying under his guidance was a great privilege and honor. I am extremely grateful for what he has offered me.

I am extremely grateful to my parents for their love, prayers, caring, and sacrifices for educating and preparing me for my future, for their love, understanding, and continuing support in completing this research work.

I express my special thanks to the Head of the Software Department, Faculty of Information Technology, University of Babylon, Prof. Ahmed Saleem Al-Saffar, and all professors of the Software Department for all their advice that helped me and their genuine support throughout this research work. I also thank all the staff of the Software Department, Information Technology Collage, Babylon University.

Thesis Related Publication



The 5th International Conference on Information Technology, Applied Mathematics and Statistics

ICITAMS 2023

Dewaniyah, Iraq

FORMAL ACCEPTANCE AND INVITATION LETTER

From

International Conference on Information Technology, Applied Mathematics and Statistics 2023

Paper ID: 1570878160

Date: 10th March 2023

Dear respected author(s):

Alyaa Abdual Kahdum
University of Babylon, Iraq

Wafaa AL-Hameed
University of Babylon, Iraq

We are delighted and pleased to inform you that your research paper entitled (***Unsupervised Automated keyphrase Extraction Approaches: A Literature Review***) has been reviewed and accepted by ICITAMS 2023 committee for oral presentation at ICITAMS 2023 conference 20-22 March 2023 and going to be submitted to the IEEE Xplore digital library.

We would like to thank you for your submission and participation.

Dhiah

Asst Prof. Dr. Dhiah Al-Shammary
University of Al-Qadisiyah
Conference Chair

UNIVERSITY OF AL-QADISIYAH

College of Computer Science and Information Technology

Dewaniyah

Iraq

Email: csit@qu.edu.iq

Website: www.qu.edu.iq/ICITAMS

ICITAMS 2023

Abstract

Due to the technical improvements and the exponential growth of textual data and digital sources, there is a challenge to extract high-quality Automatic keyphrase extraction (AKE) is crucial to many NLP and information retrieval tasks, including document summarizing, classification, article recommendation, and full-text indexing. The aim of this work is to obtain a semantic understanding of the query and index documents by using the embedding technique to improve the performance of IR systems.

The proposed method consists of several stages. The first stage represents collecting documents in one file after decompressing the dataset folder using the Linux system and specifying only documents with a decision for use within the system. The second stage involves performing several pre-processing steps on documents. Then an unsupervised keyphrase extraction method is implemented, and the extracted candidate key phrases undergo the applying a universal Sentence encoder (USE) to produce embedding vectors with keeping the most informative using Maximal Marginal Relevance (MMR). The step of keyphrase extraction was evaluated using the (Inspec) dataset which has a file named key involve (Golden keyphrases added by experts to represent each document). The third stage involves indexing and ranking using two approaches, the first method average keyphrase embedding vector to obtain a document score, and the second method constructs an inverted index to compute and rank the most similar ones with query embedding vector. The fourth stage was scoring the most documents relevant to the query vector.

The proposed retrieval model implement on the (Fire2011) dataset. The final stage was evaluating the results of the baseline and the results of the two approaches (indexing and ranking) by using mean average precision (MAP). The result of the baseline was

0.61, while the result Document Average Embedding approach was 0.5589172. It was observed that the best result was achieved with Inverted Index was 0.6277519 .

.

Table of Contents

Table of Tables

Table of Figures

Table of Algorithms

Table of Abbreviations

Abbreviations	**Descriptions**
AKE	Automatic key-phrase extraction
AP	Average precision

CBOW	Continuous Bag-of-Words
DAN	Deep averaging network
FN	False Negative
FT	False Positive
IR	Information Retrieval
MAP	Mean Average precision
MMR	Maximal Marginal Relevance
NBOW	Neural Bag-of-word model
NP	Noun Phrase
POS tagger	Part of speech tagger
TN	True Negative
TP	True Positive
USE	Universal sentence encoder
VSM	Vector space model

Chapter One: Introduction

1.1 Background

Information. retrieval. (IR) is the process of obtaining information from a collection of documents or data sources, typically in response to a user's information need. It involves searching for and retrieving relevant information based on a query or set of keywords provided by the user.

Information. retrieval. aims to provide the user with a set of documents or resources relevant to their query or information need. This is often accomplished using search engines, which use algorithms and techniques to rank and present the most relevant results to the user[1].

In recent years, it has been observed that data expansion has increased significantly, creating new obstacles for academics as they try to come up with creative ways to extract important information faster. Several strategies are used to obtain crucial data from the repository. To handle the high data speed, a variety of methods are being investigated, from storage to information retrieval[2]. Finding the needed information without the IRS was nearly impossible because of the size of the search field. Today's computer user cannot envisage obtaining the information required daily without the aid of some sort of information retrieval tool. The most popular tool for information retrieval is a web search engine, which is used to identify information sources that are relevant to the user[3]. The collection's helpful materials are indexed by a search engine, which then looks through those indexes for relevant information[4].

Research in information retrieval concentrates on efficiency in terms of query response time and storage space for indexes in a distributed setting, as well as on personalized search where outcomes may be filtered and tailored to the user's area of interest, taking into account , for example, time, geographical knowledge, and the user's prior data[1].

Typically, document representation depends on a document's weighting requirements to show its importance in the document. The fundamental technique for document representation is a vector space model (VSM) based on word2vec [5]. Additionally, high semantic levels of document representation rely on individual words, phrases, or paragraphs.

The main problem with VSM [6]is that semantic representation is missing. Word vectors (Word2vec), which may represent information such as the semantics and syntax of words, have emerged in text mining and related domains as a result of the development of neural network language models in recent years[7]. This thesis proposes an efficient method for scoring documents that are represented by embedding the most semantic keyphrases (expressing a certain part of the documents)to enhance document retrieval.

The ideas behind phrase embedding and word embedding are extremely similar. The embedding represents a phrase or a sentence's semantic meaning rather than a single word's semantic meaning[8].

A random sequence of words is represented by multi-word embeddings as a fixed-length semantic vector in a continuous vector space . This thesis makes use of the deep averaging network (DAN) model, which first averages the input word and bigram embeddings before sending them via a feed-forward deep neural network to produce sentence embeddings[9].

Keyphrases linked to a document often give a high-level description of the document and can facilitate effective information processing. Given today's massive document collections, these key terms are critical for searching and retrieving relevant information. Unfortunately, these notions are not always easily available and must be extracted from the various details in documents [10].

One advantage of employing semantic steps in IR is that it overcomes the constraints of approaches based on basic lexicographic word matching, i.e., simple IR models consider a document to be relevant according to a query only if the terms provided in the query appear in the document. By doing a syntactic search, semantic measurements allow the meaning of words to be considered. As a result, they are employed to improve old models [11]. The quality of the rules for phrase association influences the results. Therefore, This study presented a document ranking based on unsupervised automated keyphrase extraction(AKE) that can identify multi-word keyphrases and can handle polysemous words (words with multiple meanings) by using the context of the sentence in which the word appears.

1.2 Research Problem

The amount of data on the Internet is enormous and constantly expanding. There hasn't been comparable technological progress in the methods for retrieving relevant information to go along with this unrestrained information increase. The main issue with this thesis is that not all users demands may be met by the results of the **query search**. This might be attributed to the users ignorance of the document collection. As a result, some of the relevant information that was recovered may have been lost due to a lack of phrase context.

This implies that the terminology used to describe concepts in questions differs from the terminology used to describe concepts in data. Also, because of found word indexing for all documents, search, and retrieval times are slow.

1.3 Research Questions

- How might the document ranking effectiveness be improved?
- Is it possible to increase the number of relevant documents?

1.4 Thesis Objectives

This study seeks to improve retrieval system effectiveness by increasing the number of pertinent documents. The following objectives must be attained to achieve this goal:

- Extracting the candidate keyphrases and study their effectiveness in the documents ranking which help enhance returning most relevant documents.
- Embedding the query and keyphrase into the same continuous vector space to compute the semantic relatedness among text fragments by using similarity measures in that feature space.
- Implement maximal marginal relevance to decrease the redundancy and keep the most informative key phrases.
- Construct an efficient inverted data structure for indexing to compute and rank the most similar ones with query embedding vector and finally scoring the most documents relevant to the query vector

1.5 Research Significance

This study builds on the previous literature on the use of information retrieval strategies that were based on the word regardless of context. The use of semantic similarity in IR to overcome the shortcomings of methods that concentrate on matching simple lexical terms. The contribution of this study was accomplished throught the sentence context to determine the correct meaning of the word within the phrase by using an embedding vector to find the relationship between words in this phrase, which improved the traditional models.

1.6 Research Challenges

There are many challenges in the work.

- The first challenge is the inaccessibility of global data, which major search engines rely on to conduct research and evaluate the system.
- The dataset has been compressed and encrypted. As a result, this dataset's decompression and decryption were performed on a Linux system.
- It was also difficult to retrieve documents with judgments for each inquiry from different folders inside of an encrypted file that contained thousands of documents and a wide range of articles.

1.7 Related Work

Finding documents that are related to information retrieval is common, and several types of research have been done to increase the efficiency of information retrieval. Embedding is one of the modules used to extract keyphrases to improve information retrieval processes. Here, some of the previous studies dealing with these concepts are reviewed. Table 1.1 displays a list of related works. The closest studies to our work are [12],[14], and [18].

1.7.1 Keyphrases Extraction

Ajallouda, Fagroud, Zellou, et al.[12], presented the KP-USE is an unsupervised approach for key-phrase extraction from documents that uses a pre-trained Universal Sentence Encoder (USE) model to generate embeddings for the sentences in the document. divide the text into five sections and weight each one according to how semantically close it is to the document and not calculated for all section by Maximal marginal relevance.

Zhang, Huan Liu, Suge Wang, et al.[13],presented a word embedding-based keyphrase extraction technique for texts. In particular, in the first build a heterogeneous text graph embedding model to combine local context information of the word graph (i.e., the local word collocation patterns) with certain key attributes of candidate words and edges of the word graph. Next, using these learned word embeddings, a unique random-walk-based ranking algorithm is developed to rate candidate words. In order to score phrases for the purpose of choosing the top-scoring phrases as key phrases, a novel and generic phrase-scoring model based on word embeddings is provided.

Bennani-Smires, Kamil, et al.[14], presented EmbedRank, an innovative unsupervised technique that makes use of sentence embeddings, may be used to extract keyphrases from single documents . the extracted keyphrases were based on the use of sentence embedding by model Sen2Vec and not the use of sentence embedding by Universal Sentence Encoder(DAN)

Zhang, Linhan, et al.[15], presented, MDERank (Masked Document Embedding Rank) is an unsupervised keyphrase extraction approach that utilizes masked document embedding ranking. The document is mapped to its corresponding embeddings using pre-trained embedding models such as BERT or RoBERTa(embedding generation). MDERank may not be able to capture the nuances of the language and may not identify the most relevant keyphrases.

1.7.2 Information Retrieval

YE, Xin, et al.[16],presented, a method for using word embeddings to compute document similarities for improved information retrieval in software engineering. The method is designed to address the challenge of retrieving relevant software engineering documents from a large corpus of unstructured text data. The proposed method first learns word embeddings using a neural network language model, similar to Word2Vec. The word embeddings are then used to compute the cosine similarity between each pair of documents in the corpus. The resulting document similarity matrix is then used to rank the documents in response to a query, using a variety of ranking algorithms, including BM25, TF-IDF, and PageRank.

NIYOGI, Mitodru; et al.[17], present a method for learning multilingual embeddings by jointly training monolingual and cross-lingual embeddings using topically aligned corpora. The method consists of two steps: first, the authors train monolingual embeddings for each language separately using a skip-gram model, and then they use a parallel corpus to align the embeddings across languages. The aligned embeddings are then used to train a cross-lingual similarity function that can be used to retrieve relevant documents across languages. One of the strengths of the paper is its use of topically aligned corpora to learn the multilingual embeddings. This approach allows the method to capture the semantic relationships between words in different languages that are relevant to the topic of interest, improving the quality of the embeddings. Another strength is the use of a cross-lingual similarity function to retrieve relevant documents, which allows the method to capture the semantic similarity between documents across languages.

AWAD, Hadeel M[18], presented a method for incorporating semantic relevance feedback into the information retrieval process, the type of semantic linkages between the meanings of two words serves as a measure of semantic similarity, which is based on word meaning. By using the WordNet-based similarity method, it is possible to determine how semantically similar two words or sentences are.

The goal of query enrichment is to improve the semantic relationship between the query and the results by including terms that are semantically related to these senses. This is done by using Word sense disambiguation (WSD) techniques to make the word and the results more semantically related to the query.

Table 1.1: Summary of Related Works.

No	No. of ref.	Method	Year	Dataset	Evaluation Measures	Results
		Keyphrase Extraction				
1.	[12]	KP_USE	2022	• NUS • Krapivin2009 • SemEval2010	F1-Score metric for top5(F1@5	• 0.07 • 0.10 • 0.15
2.	[13]	Automatic keyphrase extraction using word embeddings	2020	• KDD • WWW • SIGIR	P,R,F1,MRR Metric for top@k,k=2	• 22.88, 11.35, 15 ,36.3 • 22.12 ,9.07 ,12. ,33.65 • 25.63 ,13.44, 17 ,40.89
3.	[14]	EmbedRank	2018	• Inspec • DUC2001 • NUS	P, R, F1 metric for top5	• 41.49 ,25.40 ,31 • 30.87, 19.66 ,24 • 3.88, 1.68, 2.35
4.	[15]	MDRank	2021	• Inspec • SemEval2017 • SemEval2010 • DUC2001 • Krapivin • NUS	• F1@K, K=15 •AvgRank	37.43 37.52 20.69 26.28 13.58 17.95 2.00 (±1.00)

					(STD) On all six benchmarks	
Information retrieval						
6.	[16]	word embeddings	2016	Wikipedia	MAP	0.248
7.	[17]	Cross-lingual embedding	2018	• FIRE2010 • FIRE 2011 • FIRE2012	MAP	• 0.1759 • 0.2259 • 0.2774
8.	[18]	Query expansion with WordNet-based similarity method	2020	FIRE 2011	MAP @10	0.61
9.	Proposed System	Extract Keyphrase using USE and MMR method then using for documents ranking	2023	FIRE 2011	MAP@10	0.6283

1.8 Thesis Organization

The thesis consists of five chapters. The rest of the chapters is as follows:

• **Chapter Two**: Presents an overview of keyphrase extraction and the main architecture of the Universal sentence encoder embedding. It also clarifies the evaluation measure for the proposed system algorithm.

• **Chapter Three**: Explains the proposed system and its algorithms.

• **Chapter Four**: Shows the outcomes of the system proposed and discusses the

evaluation of the performance of the system.

• **Chapter Five**: Demonstrates the system conclusion and possible future works to .

Chapter Two: Theoretical Background

2.1 Introduction

This Chapter mainly highlights unsupervised keyphrase extraction, embedding (word, sentence) models, Maximal marginal relevance method(MMR), Inverted index, Document Ranking, Finally, this chapter explores the theoretical background of metric output and techniques of evaluation that advance the effectiveness of the information retrieval system.

2.2 Keyphrase Extraction

The extraction approach is mining about most important words (key-terms) from the document without using a vocabulary list the words taken directly from the document, key-term is a keyword or key phrase, and the difference between them (keyword is a single word term like (web, service) while keyphrase is a multi-word lexeme like(parallel computing, and web services)). Single words used as index terms can occasionally cause misunderstandings. When employed as individual indexing keywords, the component single words in phrases like "hot dog," for instance, do not have their usual connotations and are therefore extremely deceptive. Moreover, keyphrases that are chosen from a controlled vocabulary lessen the issues that come with polysemy and synonymy in natural language[20].

2.2.1 Unsupervised Keyphrase extraction

Unsupervised key phrase extraction is automatically identifying important phrases or terms from a given text corpus without relying on pre-labeled training data[21]. The benefits of unsupervised keyphrase extract methods are applicable to texts written in any language. They do not rely on language-specific rules or dictionaries, making them versatile for multilingual text analysis, can handle large volumes of text data efficiently, and allow for the identification of domain-specific terminology and key phrases that may not be present in standard dictionaries, Unlike supervised methods that require manually annotated training data, unsupervised approaches eliminate the need for expensive and time-consuming labeling efforts[20]. This makes it easier and more cost-effective to apply keyphrase extraction to new domains or when labeled training data is limited, also extracted key phrases can provide condensed representations of documents, aiding in summarizing text[22] or improving search engine results by matching user queries with relevant key phrases, and also key phrases act as important features for grouping similar documents together[23].

Finally, unsupervised key phrase extraction methods offer a flexible and efficient way to automatically extract important phrases or terms from text data, enabling various downstream applications in information retrieval, text mining, and document analysis.

2.3 Embedding

An embedding, also known as a word embedding or a phrase(sentence) embedding, is a technique used in natural language processing to represent words or phrases as a dense vector of numerical values. The purpose of embedding is to create a suitable format for machine learning algorithms to process the semantic and syntactic information about words and phrases[24].

In NLP, embedding is a potent method that has transformed the discipline by allowing machine learning algorithms to handle natural language data with efficiency. Embedding allows algorithms to process text in a way that captures semantic and syntactic information, allowing them to carry out a variety of NLP tasks with high accuracy. Words and phrases are represented as dense vectors[25].

2.3.1 Word embedding (word2vec) in Information Retrieval

In order to increase the relevance and accuracy of search results, word embeddings have also been used in information retrieval. Words are represented as sparse vectors of one-hot encoded values in conventional information retrieval models, such as the vector space model, which do not take into account the semantic relationships between words. This can result in problems like polysemy (where a word has multiple meanings) and synonymy (where two words have the same meaning), which can produce false search results[16].

A well-liked neural network-based model for creating word embeddings is Word2Vec. The model is trained on sizable text corpora and discovers how to map each word in the vocabulary to a dense vector of numerical values, typically 100 to 300 dimensions[26].

The foundation of Word2Vec is the notion that a word s meaning can be deduced from the context in which it appears[27]. A large corpus of text is used as the model s input, and training examples are generated using a sliding window method. The model forecasts the likelihood of each target word in the window for each context word in the vocabulary, or the opposite[28].

Other information retrieval tasks, like document clustering and topic modeling, have also made use of word embeddings. When performing these tasks, word embeddings can be used to distinguish between documents that are similar by their semantic content rather than just their lexical similarity[24].

The Continuous Bag-of-Words (CBOW) model and the Skip-gram model are the two variations of Word2Vec.

1. Continuous Bag-of-Words (CBOW)

The Continuous-Bag-of-Words (CBOW) model[29], which infers the center word from the context in which it is surrounded. In equation 2.1[30] .

$$\mathrm{P}(W_i | W_{i-c}, W_{i-c+1}, \ldots\ldots, W_{i-1}, W_{i+1}, \ldots\ldots, W_{i+c-1}, W_{i+c}) \ldots .(2.1)$$

where W_i is a word at location i and c is the window size, this model maximizes the likelihood that a word will be in a given context.

As a result, a model that depends on the distributional closeness of words is produced.

In the following explanation, them concentrate on the initial iteration.

They assume W is the vocabulary set containing all words. The CBOW model trains two matrices:

(1) An input word matrix denoted by V$\in R^{N\times|W|}$, where the i^{th} column of V is the N-dimensional embedded vector for input word v_i;

(2) an output word matrix denoted by U $\in R^{|W|\times N}$, where the j^{th} row of U is the N-dimensional embedded vector for output word u_j.

In order to embed input context words, they first use the one-hot representation for each word and then use V^T to obtain the matching word vector embeddings of dimension N. They create a score vector by applying U^T to a word vector of input, and they turn that score vector into a W-dimensional probability vector by applying the softmax operation. As a result of this procedure, a probability vector will be produced that corresponds to the vector representation of the output word. The CBOW model is created by minimizing the cross-entropy loss between the probability vector and the embedding vector of the output word. This is achieved by minimizing the following objective equation 2.2[30]:

$$M(u_i) = -u_i^T \hat{v} + \log\sum_{j=1}^{|W|} \exp(u_i^T \hat{v}) \quad \ldots\ldots\ldots(2.2)$$

where $\hat{v}$ is the average of the embedded input words and u_i is the i^{th} row of matrix U. The initial values for the matrices V and U are chosen at random. Depending on the various application circumstances, the dimension N of word embedding can change. It typically has 50 to 300 dimensions. Once both matrices V or U have been obtained, they can either be combined or averaged as shown in Figure 2.1 to produce the final word embedding matrix[30].

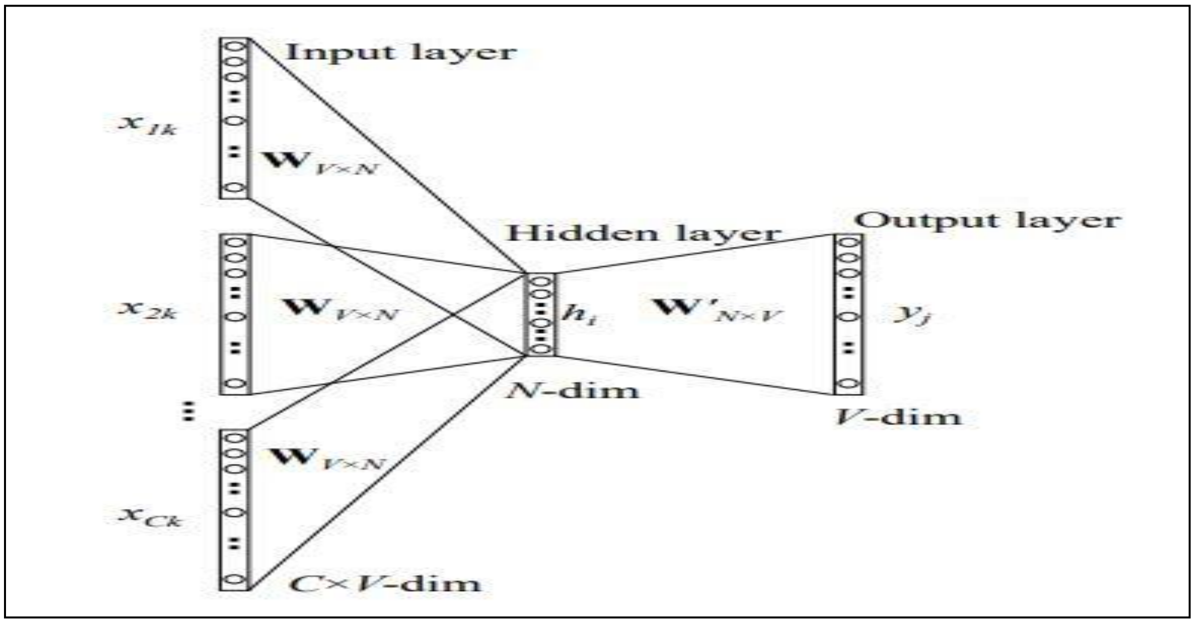


Figure 2.1: Continuous bag-of-word model(CBOW)[31]

2. Skip-gram model

The skip-gram model provides a center word and predicts the words in the surrounding context as shown in Figure 2.2. It aims to maximize the probability of context words given a particular center word and is denoted by the formula 2.3[30] :

$$\mathrm{P}(W_{i-c}, W_{i-c+1}, \ldots\ldots, W_{i-1}, W_{i+c-1,} W_{i+c} | W_i) \ldots.. \quad (2.3)$$

The optimization process is the same as for the CBOW model, but with the context and center words in the opposite order. The softmax function, which was previously stated, is a technique for producing probability distributions from word vectors. It can be written as

$$\mathrm{P(wc|wi)} = \frac{\exp(v_{wc}^{T}\ v_{wi})}{\sum_{w=1}^{|W|} \exp(v_{w}^{T}\ v_{wi})} \ldots \ldots \ldots \ldots \ldots. \qquad (2.4)[30]$$

This softmax function is not the most effective one because, in order to normalize it, we must add up all W words. Additionally, hierarchical softmax and negative sampling are more effective. The softmax model's log probability is maximized via the negative sampling technique by summing only a smaller portion of the *W* words. By analyzing only log2 *W* words, hierarchical softmax also approaches the entire softmax function. The output layer of hierarchical softmax is represented as a binary tree, where the words are the leaves and each node stands for the relative probabilities of its child nodes. These two methods are effective at identifying intricate linguistic patterns and making predictions for little context windows. However, it might be further enhanced if global co-occurrence statistics are used[29].

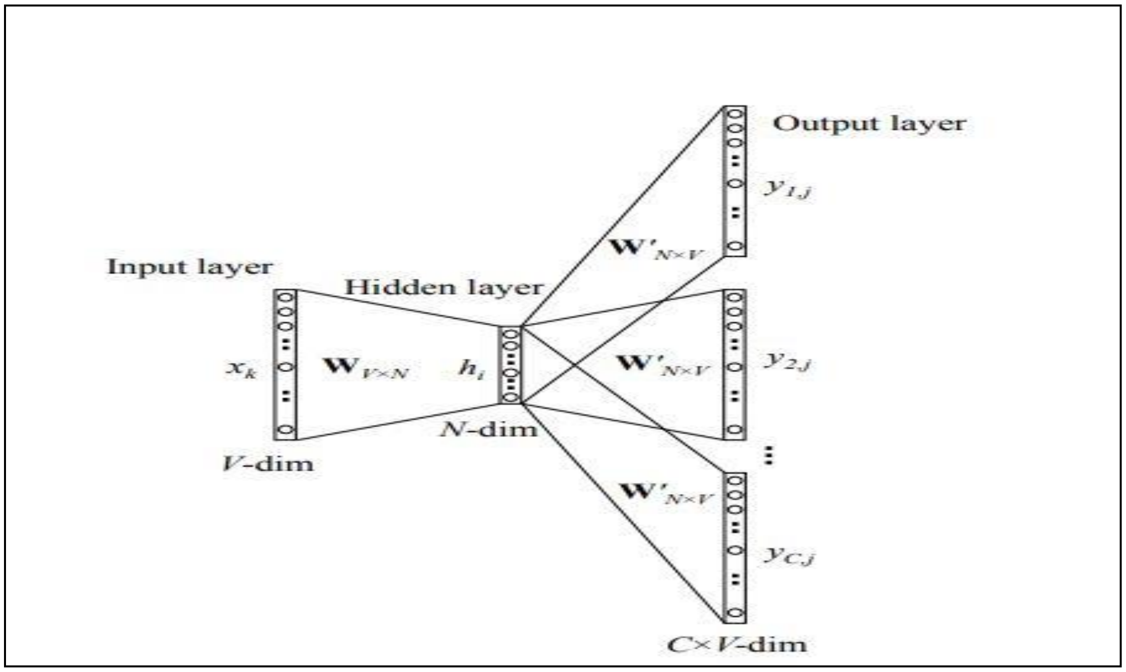


Figure 2.2: The skip-gram model [31]

Overall, word embeddings have the potential to enhance the precision and relevance of information retrieval systems and to make it possible to process natural language data more successfully in a variety of contexts.

2.3.2 Sentence embedding (USE) in Information Retrieval

Sentence embedding is the process of representing a sentence. as a dense vector of fixed length, usually by means of a neural network[32]. Once the sentence is encoded as a fixed-length vector, it can be used to compute similarity scores between sentences or to perform other NLP tasks. For example, cosine similarity can be used to measure the similarity between two sentence vectors, and a classification model can be trained on top of the sentence embeddings to predict the label of a given text[12].

In IR, there are various methods for creating sentence embeddings. Pre-trained language models are a popular approach. One of common models is the

Universal Sentence Encoder (USE), which use deep neural networks to encode sentences into fixed-length vectors that accurately reflect their semantic meaning[33]. These models explained robust transfer task performance rather .than models which used pre-trained word embedding like the word2vec model which was produced by[34]. Because word embeddings are especially effective .in cases where there is limited training data, leading to .sparsity and poor vocabulary coverage, which in turn leads to poor generalization .capabilities[35] . This thesis used a Universal Sentence Encoder(USE) model to get sentences embedding vectors.

The Universal Sentence Encoder (USE) is a neural network-based model developed by Google that generates high-quality embeddings for natural language sentences. It was introduced in 2018[33] and has been widely used in a variety of natural language processing (NLP) applications.

The Universal Sentence Encoder (USE) includes two different models: one based on a deep averaging network (DAN) architecture, and another based on a transformer architecture. The DAN-based USE model is a simpler architecture that computes a fixed-length sentence embedding by averaging input embeddings of the individual words in the sentence and bi-grams before passing them through a feed forward deep neural network(DNN)to produce sentence embeddings. This model is faster to train and generally more computationally efficient than the transformer-based model. The transformer-based USE model, on the other hand, is a more complex architecture that uses a self-attention mechanism to encode the input sentence into a fixed-length embedding(512-dimensional)[33]. This model is slower to train and requires more computational resources, but is generally considered to produce higher-quality sentence embeddings than the DAN-based model. The Deep Averaging network. (DAN)encoder will be. discussed in detail because this. architecture has been. used in this work.

1. Deep Averaging network (DAN) encoder

This encoder is built using the architecture suggested by Lyyer and colleagues [36]. The deep averaging network (DAN) has three basic phases, as outlined by the authors Lyyer and et al.

First phase: Compute the vector average of embeddings associated with the tokens sequence input.

Second phase: Take the average and run it through one or more feed forward layers.

Third phase: Apply(linear)classification on the representation of the final layer.

Typically, deep feed-forward neural networks are built with the concept that each layer will learn a more abstract representation of the input than the one before it[37].DAN applied on Neural Bag-of-word(NBOW)model. Consider text classification when using the NBOW: equation 2.5 proposed by[36] is used to transform an input sequence of tokens *X* into one of *kla* labels.

$$z = g(w \in X) = \frac{1}{|X|} \sum_{w \in X} v_w \text{.........} \qquad (2.5)$$

where z represents the input text X as a vector by averaging the word vectors $v_{w\in X}$, g is a composition function, v_w is a sequence of word embeddings, and X is a sequence of tokens. When z is sent to a softmax layer, equation 2.6[36]is generated as the estimated probabilities for each output label.

$$Y^{\hat{}} = \text{softmax}(W_s \times z + b) \quad \text{………} \qquad (2.6)$$

where W_s is a matrix of kla ×dim values for a dataset with kla output labels, b is a bias term, and the softmax activation function is given in equation 2.7[36]

$$\text{softmax}(q) = \frac{\exp(q)}{\sum_{j=1}^{k} \exp(q_j)} \quad \ldots\ldots\ldots\ldots\ldots \quad (2.7)$$

Before using the softmax, they were able to further transform z by adding more layers.The writer intended to have n layers, z_1..n.They used equation 2.8 [36]to calculate each layer.

$$z_i = g(z_{i-1}) = f(W_s \times z_{i-1} + b_i) \ldots\ldots\ldots\ldots\ldots (2.8)$$

DAN starts by working on word embeddings, and any present bi-grams in a sentence are averaged. After that, they are fed into an n-layer feed-forward deep neural network to produce 512-dimensional sentence embeddings. The word and bi-gram embeddings are learned during training as showed the DAN layout in Figure 2.3 by sentence1("Predator is a masterpiece")[36]. The sentence2 ("Hello world") is an example explaining work DAN as Figure 2.4[38].

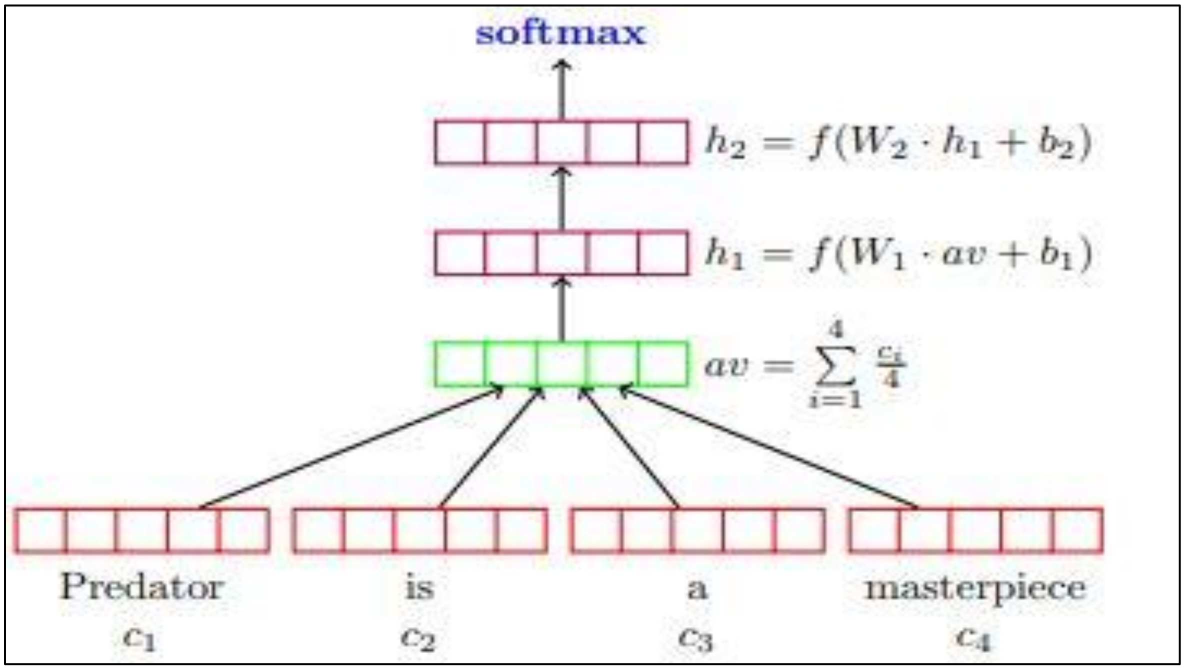

softmax
$h_2 = f(W_2 \cdot h_1 + b_2)$
$h_1 = f(W_1 \cdot av + b_1)$
$av = \sum_{i=1}^{4} \frac{c_i}{4}$
Predator
is
a
masterpiece
c_1
c_2
c_3
c_4

Figure 2.3 Layout DAN which Applied on Sentence1[36]

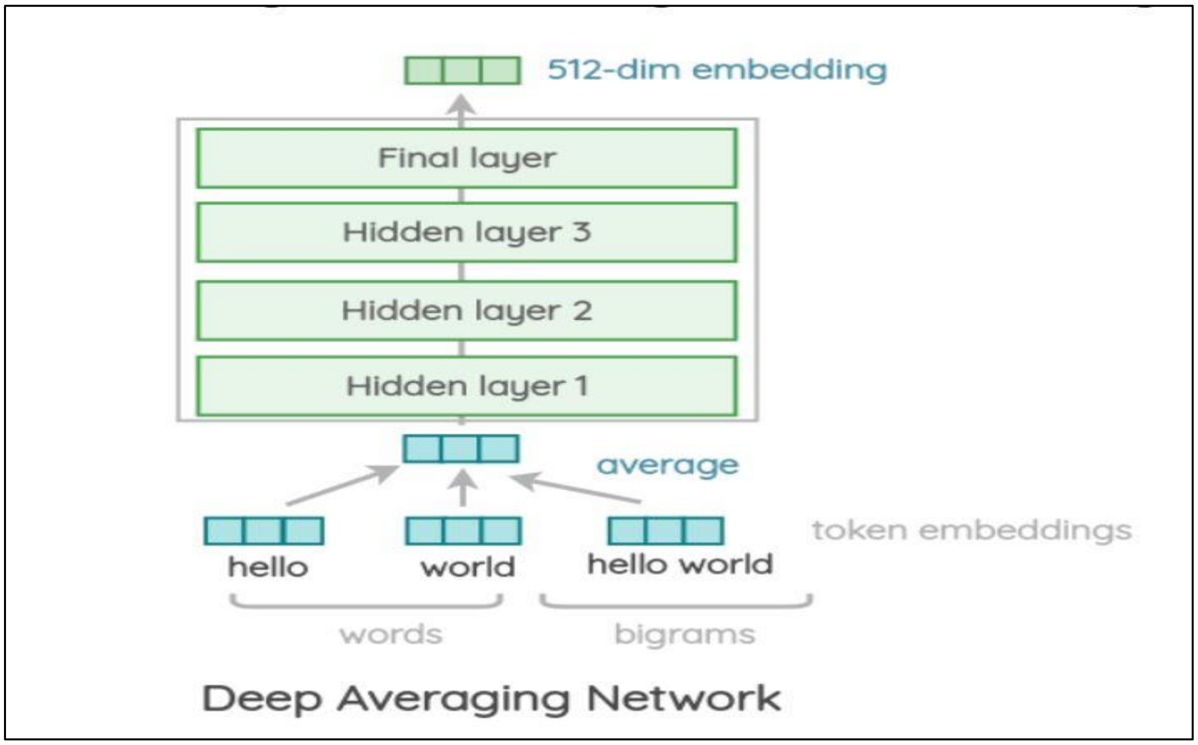


Figure 2.4 Layout DAN which Applied on Sentence2[38]

2.4 Maximal Marginal Relevance method(MMR)

Maximal Marginal Relevance (MMR) is a method used in information retrieval to rank and select relevant documents based on their similarity to a query while also ensuring diversity in the retrieved results. It was first introduced by Carbonell and Goldstein in 1998[39].

MMR from the information retrieval and text summarization is based on the set of all initially retrieved documents, R, for a given input query Q, and on an initially empty set S representing documents that are selected as good answers for Q. S is iteratively populated by computing MMR as described in equation 2.9 [39], where D_i and D_j are retrieved documents, and *Sim1* and *Sim2* are similarity functions.

$$\text{MMR:=} arg\,max_{D_i \in R \setminus S}\, [\lambda. Sim1\ \ (D_i, Q) - \ (1\lambda) max_{D_j \epsilon S}\ \ Sim2(D_i, Dj)] \ldots.(2.9)$$

While when λ = 0 MMR computes a maximal diversity ranking of the documents in R, when λ = 1 MMR computes a standard, relevance-ranked list.

Maximal Marginal Relevance (MMR) is a method that can also be applied to keyphrase extraction[14], by adapting MMR to keyphrase extraction, in order to combine keyphrase informativeness with dissimilarity among selected keyphrases, as shown in equation 2.10 [14]

$$\text{MMR:=arg}\, max_{C_i \in C \setminus K}\, [\lambda. \widetilde{cos}_{sim}\, (C_i, doc) - (1 - \lambda) max_{C_j \epsilon K}\, \widetilde{cos}_{sim} (C_i, Cj)] \ldots\ldots(2.10)$$

where Ci and Cj are the embeddings of the respective candidate phrases i and j, and C is the set of candidate keyphrases, K is the set of extracted keyphrases, the doc is the full document embedding. The following equations 2.11a, 2.11b [14]define $\widetilde{cos}_{sim}$, which is a normalized cosine similarity [40].This guarantees that the relevance and diversity components of the equation have equal weight when = 0.5.

$$\widetilde{cos}_{sim} \ (C_i, doc) = \frac{\overline{\frac{0.5+}{ncos_{sim}(C_i, \text{doc}) - ncos_{sim}(\text{C}, \text{doc})}}}{\sigma(ncos_{sim} \ (\text{C}, \text{doc}))} \ \ldots\ldots\ldots\ldots \qquad (2.11a)$$

$$ncos_{sim} \ (C_i, \text{doc}) = \frac{cos_{sim} \ (C_i, \text{doc}) - min_{C_j \in C} cos_{sim} \ (C_j, \text{doc})}{max_{C_j \in C} cos_{sim} \ (C_j, \text{doc})} \ \ldots\ldots..(2.11b)$$

2.5 Inverted Index

In Information Retrieval (IR), an inverted index is a data structure that is commonly used to facilitate efficient full-text search and retrieval of documents based on terms or keywords. It is a key component of many search engines[41].

In a traditional database, data is typically organized based on the documents or records, and each document contains its own information. However, in an inverted index, the focus is on the terms or keywords within the documents[42].

2.5.1 Dictionary

In the context of information retrieval, a dictionary refers to a data structure that stores and provides access to the terms or words present in a collection of documents. It is vital in many information retrieval systems, including search engines. The dictionary facilitates efficient searching, indexing, and retrieval of documents based on user queries[43].

A dictionary is used in information retrieval (Inverted Index) Along with the term ID, the dictionary may also store additional information, such as document frequencies (the number of documents a term appears in) or term frequencies (the frequency of a term within a specific document). This information is used to build an inverted index, which maps each term to the documents that contain it. The inverted index facilitates a quick lookup of documents based on terms present in a query[42].

2.5.2 Posting list

In information retrieval, a posting list is a data structure used to store the occurrence information of terms in a document collection or corpus. It is a fundamental component of inverted indexes, which are commonly used in search engines and other information retrieval systems.

Posting lists are used to efficiently retrieve documents containing specific terms during the search process. They allow for quick lookup of documents that contain a given term, enabling fast retrieval of relevant documents for a given query[44].

In practice, posting lists are often compressed or optimized to reduce the memory footprint and improve search performance. Techniques like delta encoding, variable-byte encoding, or even more advanced compression algorithms are applied to reduce the storage requirements of posting lists in large-scale information retrieval systems.

2.6 Document Ranking

Ranking documents is a critical step in information retrieval (IR) systems, where the goal is to retrieve a set of documents that are relevant to a given user query.

Ranking documents by embedding vector involves representing the keyphrase of documents and the query as dense vectors using a sentence embedding technique(USE)[12]. Using embedding in document ranking has several advantages over traditional methods such as the vector space model or BM25 ranking algorithm. Here are some of the benefits of using embeddings for document ranking:

Semantic similarity: Embeddings can capture the semantic meaning of words and phrases, which enables more accurate matching of queries to documents. For example, if a user searches for "car repair," an embedding-based model can retrieve documents that contain synonyms or related terms such as "auto maintenance" or "vehicle service."[24]

Efficiency: Embeddings can reduce the dimensionality of the data, which makes the retrieval process more efficient. This is because embeddings typically have lower dimensionality than the term-document matrix used in traditional ranking methods.[45]

The system returns the top matching documents in the set of documents related to a query. The documents are ranked according to scores matching the query and the document. Those scores calculate how well the match between the document and the query[46]. There is one direction to assigning a score of matching a query and document pair, using cosine similarity

1. Cosine Similarity

It is one of the most widely used methods of rating the retrieved documents by similarity or proximity between the vectors of query and document [47][48]. The cosine similarity is demonstrated in equation 2.12.

$$\text{cosine}(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{||q|| \cdot ||d||} = \frac{\sum_1^n \text{qidi}}{\sqrt{\sum_1^n q^2}\sqrt{\sum_1^n d^2}} \quad \ldots\ldots\ldots\ldots.. (2.12)$$

Where q: represents the query, d: represents the document .

$\vec{q} \cdot \vec{d} = \sum_1^n q_i d_i = q_1\ d_1\ + q_2\ d_2\ + \cdots + q_n\ d_n$ is the dot product of the two vectors.

2.7 Evaluation Methods

The normative method for evaluating the system of IR refers to the fact of relevant and irrelevant records [46]. The assessment in the IR framework is, therefore, to assess how well the program meets the users information needs. There are multiple approaches to assessing the IR system's performance of retrieval. Precision is the fundamental metrics used to evaluate the research methodology [49]. These steps are determined using The Confusion Matrix as illustrated in Table 2.1 [46].

Table 2.1: The Confusion Matrix

	Relevant	Irrelevant
Retrieved	True-Positive (TP)	False-Positive (FP)
Not retrieved	False-Negative (FN)	True-Negative (TN)

1. True Positive (TP): number of relevant documents that are retrieved.

2. False Negative (FN): number of relevant documents that are not retrieved.

3. False Positive (FP): number of non-relevant documents that are retrieved.

4. True Negative (TN): number of documents that are nonrelevant and not retrieved.

1. Precision

Precision is the number of true positive (tp) divided by the total number of true positive (tp) and false positives (fp) or the rate of relevant documents retrieved to the total number of relevant and irrelevant documents retrieved[50], as shown in Equation 2.13.

$$\text{Precision} = \frac{tp}{tp+fp} \quad \ldots\ldots\ldots \quad (2.13)$$

2. Mean Average Precision(MAP)

Depending on many query topics, MAP is carried out on an information recovery system by computing the average of each query's average precision measured value[51]. Average precision (AP) is calculated by measuring the average precision for all the relevant retrieved documents as shown in Equation 2.14[52].

$$\text{AP} = \frac{1}{n} \sum_{i=1}^{n} \text{Precision(pi)} \quad \ldots\ldots\ldots \quad (2.14)$$

Where *n*: represents the number of relevant retrieved documents pi: represents the average precision of each query .

This was achieved by measuring the mean value of multiple queries average precision . The form that measures the MAP is presented in Equation 2.15[53].

$$\text{MAP} = \frac{\sum_{i=1}^{Q} \text{AP(i)}}{|Q|} \quad \ldots\ldots\ldots\ldots \quad (2.15)$$

Chapter Three: The Proposed System

3.1 Introduction

Finding relevant information on a large website with millions of online entries or in numerous document databases might be difficult for users. To confirm that the relevant documents have been acquired, use an embedding vector to obtain a semantic relation between terms in the query and documents.

This thesis includes an approach to finding top-ranking documents by creating query embedding and document embedding vectors by extracting keyphrases (more informative) from this document.

3.2 The Proposed System Architecture

The proposed system architecture includes two main phases (Offline phase and Online phase). In the training (offline) stage, many procedures are implemented beginning with preprocessing (Lowercasing, Tokenization, Stop Words removal, and Stemming), also extracting candidate keyphrases (noun phrases) in the full document. Following the embedding technique (DAN)is carried out for excluding the redundant keyphrases using the MMR method. Then, for indexing and ranking, two approaches are considered: the first method creates an embedding document vector by normalization embedding vector rank keyphrases for this document then cosine similarity with the embedding query vector. Then build the second method for enhancement above method, through constructing an inverted index to compute and rank the most similar ones with query embedding vector and finally score the most documents relevant to the query vector. Regarding the online stage, while a query is supplied by the user, many processes are conducted (cosine similarity, Keyphrase scoring, and Document scoring) to return the most relevant documents as shown in Figure 3.1. Each stage of this system will be explained in the next sections.

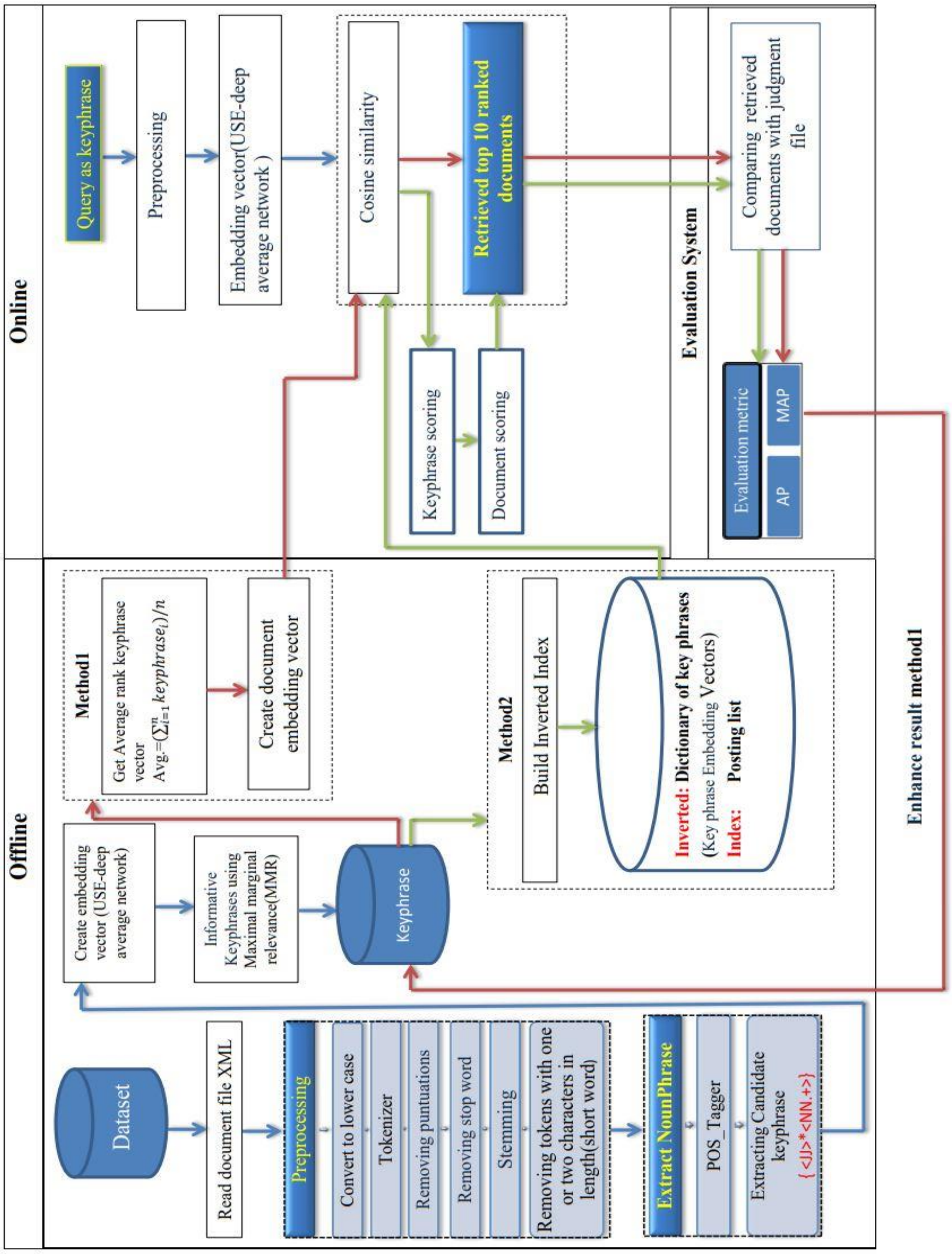

Offline
Dataset
Read document file XML
Preprocessing
Convert to lower case
Tokenizer
Removing puntuations
Removing stop word
Stemming
Removing tokens with one or two characters in length(short word)
Extract NounPhrase
POS_Tagger
Extracting Candidate keyphrase
{<JJ>*<NN.+>}
Create embedding vector (USE-deep average network)
Informative Keyphrases using Maximal marginal relevance(MMR)
Keyphrase
Method1
Get Average rank keyphrase vector
Avg.=(∑ keyphrase)/n
Create document embedding vector
Method2
Build Inverted Index
Inverted: Dictionary of key phrases
(Key phrase Embedding Vectors)
Index: Posting list
Enhance result method1
Online
Query as keyphrase
Preprocessing
Embedding vector(USE-deep average network)
Cosine similarity
Retrieved top 10 ranked documents
Keyphrase scoring
Document scoring
Evaluation System
Comparing retrieved documents with judgment file
Evaluation metric
AP
MAP

Figure 3.1: Proposed System.

3.3 Description of Dataset

For working on this thesis, we need two data sets. The first dataset is used for evaluation of the keyphrase extracted because have golden files for this keyphrase in the dataset then apply second dataset (FIRE) is used for documents ranking, a description of these datasets is in section 3.3.1 and section 3.3.2 .

3.3.1 Inspec

Inspec is a publicly available dataset that is commonly used for evaluating automatic keyphrase extraction methods in natural language processing (NLP). The dataset consists of abstracts from scientific articles in the field of computer science, information science, and electrical engineering, among others.

The Inspec dataset contains approximately 2,000 documents. The documents were selected from the Inspec database, which is a bibliographic database of scientific literature in these fields. The documents cover a broad range of topics, including information retrieval, machine learning, computer networks, and data mining[47]. As shown in the following example Figure 3.2

The importance of continuity: a reply to Chris Eliasmith
In his reply to Eliasmith (see ibid., vol.11, p.417-26, 2001) Poznanski
considers how the notion of continuity of dynamic representations
serves as a beacon for an integrative neuroscience to emerge. He
considers how the importance of continuity has come under attack from
Eliasmith (2001) who claims: (i) continuous nature of neurons is not
relevant to the information they process, and (ii) continuity is not
important for understanding cognition because the various sources of
noise introduce uncertainty into spike arrival times, so encoding and
decoding spike trains must be discrete at some level

Figure 3.2 : The structure samples of the document(Inspec)

Each document in the Inspec dataset has been manually annotated with a set of keyphrases, which represent important concepts or topics in the text. These keyphrases were chosen by domain experts and reflect the most salient and relevant information in the document. As shown in the following example Figure 3.3.

Continuity
dynamic representations
integrative neuroscience
neurons
cognition
uncertainty
spike arrival times
spike trains
cognitive systems
neural nets
cognitive systems
neural nets
neurophysiology

Figure 3.3 : The golden keyphrase

3.3.2 FIRE 2011

The data used in this system is obtained from the FIRE dataset of Forum for Information Retrieval Evaluation that is accessible on the website (http://fire.irsi.res.in/fire/static/data) and applying for access to the FIRE Information Retrieval Text Research Collection.

The suggested approach was implemented on the FIRE 2011 dataset in English. It is a category created specifically for experiments with information retrieval (IR). This collection consists of several newspaper stories on many subjects, including local news, sports, commercial, political, and medical news[48].

The documents and queries in the Fire 2011 dataset are represented using XML markup language. The XML tags are used to provide a standardized format for representing and processing the text data and associated metadata. Both documents and queries are represented using specific metadata. As shown in the following example Figure 3.4

```
<DOC>
<DOCNO>en.13.1.3.2009.6.1</DOCNO>
<TITLE> Ukhia, other Upazilas go to polls Monday </TITLE>
<TEXT>
 Dhaka, May 31 (bdnews24.com)The suspended upazila elections in Ukhia in Cox's Bazar will
be held on Monday, four months after chairmen were voted in across the country to the local
government councils. The polls, and pending re-votes in some other centres where polling was
suspended or disputed, will run through to 4pm from 8am, Mohammad Abdul Baten, senior
assistant secretary at the Election Commission secretariat, told bdnews24.com. The polls to one
centre under Jagannathpur Upazila in Sunamganj will not be held on Monday as there is a court
directive. Three candidates for chairmanship, nine for vice chairmanship and three for women
vice chairmanship will contest in the polls. The Upazila accounts for 95,899 voters. 6. The
suspended polls to two centres under Atpara Upazila in Netrokona, one at Pakundia Upazila in
Kishoreganj for chairmanship, three centres at Muradnagar Upazila for vice chairmanship will be
held on the day. The suspended polls will also be held to four centres at Chandina Upazila for
women vice chairmanship in Comilla and 10 centres at Chhagalnaiya Upazila for all Upazila
posts in Feni. The polls were suspended at Ukhia and Dighinala Upazila before Jan 22, and later
on Jan 22 the polls to Barura in Comilla, Brahmanbaria Sadar, Ramganj in Laxmipur and
Belkuchi Upazila in Sirajganj due to various irregularities. bdn
</TEXT>
</DOC>
```

Figure 3.4:The structure of document(FIRE)

The queries in the FIRE 2011 dataset are short text strings that represent the information needs of users. The structure of queries in the FIRE 2011 dataset is represented Like documents, As shown in the following example Figure 3.5.

```
<topics>
<top lang='en'>
<num>126</num>
<title>Swine flu vaccine</title>
<desc>Indigenous vaccine made in India for swine flu prevention</desc>
```

```
<narr>Relevant documents should contain information related to making indigenous swine flu vaccines in India, the vaccine's use on humans and animals, arrangements that are in place to prevent scarcity / unavailability of the vaccine, and the vaccine's role in saving lives.</narr>
</top>
  .
  .
  .

<top lang='en'>
<num>175</num>
<title>Sachin Tendulkar's record of runs in Test Cricket</title>
<desc>Sachin Tendulkar's record of runs in Test Cricket</desc>
<narr>Relevant documents should contain information about the records Sachin Tendulkar has broken including his own, in Test Cricket. Recent, new records he has created are also relevant. One-day Cricket and T-20 cricket are not relevant in this context.</narr>
</top>
</topics>
```

Figure 3.5 : The structure samples of the query(FIRE)

The document tags and the query tags facilitate the processing and analysis of the query text by information retrieval systems.

Finally, also the dataset contains Relevance judgments: A set of relevance judgments that indicate the relevance of each document to a set of predefined queries. The relevance judgments are typically represented as a binary relevance score (relevant or not relevant) or a graded relevance score (e.g., highly relevant, somewhat relevant, not relevant), As shown in the following example in Table 3.1.

Table 3.1: judgment file

Query num	Documents	relevant or not relevant
126	en.13.10.19.2009.7.30	1
126	en.13.25.179.2009.11.8	0

126	en.13.25.215.2009.11.8	0
126	en.13.27.301.2009.11.20	1
. . .		
127	en.15.160.419.2010.1.15	1
127	en.15.181.411.2010.5.14	0
127	en.15.201.270.2007.8.12	1
. . .		
175	en.3.382.145.2007.7.29	0
175	en.3.390.323.2007.9.6	1
175	en.3.407.353.2007.11.25	1

The dataset details are described in Table 3.2

Table 3.2: Dataset Details[54][18]

Size	234.4MB
Language	English
Type	News article
Total Number of documents	89.286
Selected Number of documents	5000
Average number of sentences per document	100
Average sentence length (in words)	22
Number of queries	50
Query ID	126-175
Average length of Queries	3.34 words
Average length of document	252.15 words
Minimum length document	261 words
Maximum length document	374 Words

3.4 Preprocessing Stage

Pre-processing of the data is **an initial stage of the proposed system. In order to clean the documents and queries from the unimportant** and unwanted data, they should be pre-processed. Within this thesis, several pre-processing approaches have been applied.

3.4.1 Lowercasing

Data lowercasing is one of the easiest and most effective ways of the preprocessing stage of data. It is applied to most. information extraction and NLP issues and can help the consistency of expected output significantly. Table 3.3 illustrates how sparsity problems are solved by lowercasing, where the same term in lower case for different situations identifies terms.

Table 3.3: Converting terms into lower case format

Word	**Lower cased**
Stanford sTanford stanforD	Stanford

3.4.2 Tokenization

Tokenization is a process of dividing the text into tokens or sentences. This process removes spaces between words, dots, punctuations, and special characters to split the text into tokens. Examples of special "characters are (!,@,#,<,>,:,?,(,),*,&,|).

3.4.3 Stop Words removal

Stop words are a collection of words widely used in one language. The idea behind using stop words is that focusing on the relevant and important terms by eliminating low-information words from the” text. Table 3.4 illustrates a set of stop words.

Table 3.4: Some of Stop Words

A	ALTHOUGH	ANY	BECAME	DO	ELSE	FOR
ABOUT	ALWAYS	ARE	BUT	DOSE	EITHER	FROM
ACTUALLY	AM	AS	CAN	EACH	HAVE	ALMOST
AN	AT	COULD	ITS	I	ALSO	AND
BE	DID	IS	IT	WE	OUR	YOU
HIM	HIS	SHE	HE	WILL	SHOULD	THEM

3.4.4 Stemming

Stemming normally removes the suffix letters and returns the words to the root by implementing a set of rules that actually rely on the long matching. This method decreases the size of a dictionary that includes all the terms in the document's set by reducing words with identical meanings and have different forms to the same form. There are various algorithms for stemming. Porter'sAlgorithm is the most common algorithm, defined also for being effective for English[55]. Table 3.5 illustrates an example of stemming with Porter Stemmer.

Table 3.5: Example of stemming

Original word	Stemmed word
Wait	Wait
Waiting	Wait
Waited	Wait
Waits	Wait

The steps of preprocessing for each document and query are illustrated in Algorithm 3.1 .

Algorithm 3.1: Pre-processing of data

```
Input: d,q     //d :represent document ,q:represent query
Output: preprocessed text
Begin
    Step1: For each document in the  collection
    Begin
        Step1.1: Convert the terms of the  document into lowercase.
        Step1.2: Splitting the document  into tokens
        Step1.3: Removing punctuations  from tokens.
        Step1.4: Removing stop words  from tokens.
        Step1.5: Removing suffix for each  term and return to the root    form.
        Step1.6:Return preprocessed text.
    End for
```

3.4 Extract Noun Phrase

Extract candidate key phrases by extracting noun phrases based on Regular expression with a custom grammar rule to match zero or more adjectives followed by one or multiple nouns depending on a Part-of-Speech (POS) tagger.

3.4.1 Part-of-Speech (POS) Tagger

A Part-of-Speech (POS) tagger is a tool or algorithm that automatically assigns a grammatical category or part-of-speech tag to each word in a given text, such as noun, verb, adjective, adverb, preposition, pronoun, conjunction, interjection, and so on. The process of POS tagging involves analyzing the context and syntactic structure of each word in a sentence or document and then assigning the most likely tag based on the word's role and function in the sentence. This is typically done using statistical models or rule-based approaches that take into account factors such as the word's position in the sentence, its morphology, and its surrounding words.

There are many POS taggers available, both open-source and commercial, that use a variety of techniques and algorithms to perform this task. Some popular POS taggers include the Stanford POS Tagger, NLTK (Natural Language Toolkit) POS Tagger, and spacy POS Tagger. These tools are often trained on large corpora of annotated text to improve their accuracy and performance. E.g. ("Tokenization is the process of breaking down a large text into smaller chunks called tokens."). Table 3.6 represents the words and corresponding parts of speech.

Table 3.6: Part of Speeches

Words	Part of speech
Tokenization	NN(noun)
Is	VBZ(verb, 3rd person sing. present takes)
The	DT(determiner)
process	NN(noun)
Of	IN(preposition/subordinating conjunction)
breaking	VBG(verb, gerund/present participle taking)
Down	RP(particle give)
A	NN(noun)
Large	JJ(adjective)
Text	NN(noun)

Into	IN(preposition/subordinating conjunction)
smaller	JJR(adjective, comparative)
chunks	NNS(noun plural)
called	VBD(verb, past tense)
tokens	NNS(noun plural)

3.4.2 Extract candidate keyphrases

Extract the key phrase in the form (NP: { <JJ>*<NN.+>})using Part of speech tagging (POS) from original query and each title of the documents in the corpus . key phrases that consist of zero or more adjectives followed by one or multiple nouns. E.g (A molecular equivalence number (meqnum) classifies a molecule with respect to a class of structural features or topological shapes such as its cyclic system or its set of functional groups. Meqnums can be used to organize molecular structures into nonoverlapping, yet highly relatable classes.). Table 3.7 represents the candidate keyphrase and corresponding parts of speeches.

Table 3.7 Represents the candidate keyphrase and corresponding parts of speeches.

Candidate keyphrase	parts of speeches
molecular equivalence number (meqnum)	('molecular', 'JJ'),('equivalence', 'NN'), ('number', 'NN'), '(meqnum)', 'NN')
structural features	('structural', 'JJ'), ('features', 'NNS')
functional groups meqnums	('functional', 'JJ'), ('groups.', 'NN'), ('Meqnums', 'NNS')
cyclic system	('cyclic', 'JJ'), ('system', 'NN')
Relatable classes	('relatable', 'JJ'), ('classes', 'NN')

The steps of extracting candidate keyphrases for each, document are .illustrated in Algorithm 3.2.

Algorithm 3.2: Extract candidate keyphrase from data

```
Input: Text Files doc of Dataset
Output: list of candidate keyphrases
Begin
    Step1: For each document
        Begin
        Step1.1: Preprocessing document      // using Algorithm3.1
        Step1.2: Apply part of speech (POS)
        Step1.3:grammer="NP::{<JJ>*<NN.+>}"//Define the regular expression
        Step1.4:If POS tagger equal to grammer // Extract candidate key-phrases
                    For each sentence in tagged data
                        For each subtree in subtrees //Build tree from parse sentence
                            Candidate = a concatenation of words that is postagger in
                            leaves subtree
                            list of candidate keyphrases = Candidate
                        End
                    End
     End
     Step 2: Return list of candidate keyphrases
End
```

3.5 Embedding distributor

An embedding distributor is a component responsible for distributing word embeddings or vector representations of words or phrases. In the context of extracting key phrases, an embedding distributor can be used to calculate and distribute embeddings for individual words or phrases(sentence). In Sentence embedding using a pre-trained sentence embedding model USE(deep average network) to convert each sentence into a dense vector representation, illustrated in the algorithm 3.3.

Algorithm 3.3: Sentence embedding using Deep average network

```
Input:
X: Candidate keyphrase (sequence of token)  //where dim is the dimension of the word embeddings, and word embeddings in X are already pre-trained.
Y: Target sentence embedding(USE).
Output: Candidate keyphrase vector fixed length 512 dimensions.
Begin
  Step1://Initialize the model parameters
     • Input dimension: The dimensionality of word embeddings.
     • Hidden dimension: The desired dimensionality of the hidden layer.
     • Output dimension: The desired dimensionality of the sentence embedding.
     • Learning rate: The rate at which the model parameters are updated during backpropagation.
     • Num_epochs: is the number of training epochs.
     • batch size: the number of training examples used in one forward and backward pass of the neural network during the training process
     • W1: Weight matrix for the connection between the input layer and the hidden layer.
     • b1: Bias vector for the hidden layer.
     • W2: Weight matrix for the connection between the hidden layer and the output layer.
     • b2: Bias vector for the output layer.// W and b with small random values
  Step2:For each token  in X:
```

```
Begin
    Step 2.1 : E=[e($X$($token$))] //create embedding token by using pretrain cbow model
    Step2.2: Compute averaging the word embeddings in $X$[$token$].
        av=(summuation($X_{token}$))/no.of $token$
End
Step3:For epoch in Num_epochs:
Begin
    // Pass the average embedding through one or more fully connected layers.
    Step 3.1: Compute the hidden layer activations h1 = relu($W1$ * av + $b1$)
    // Compute the output layer
    Step3.2: Compute the output $Y\hat{}$ = softmax(W2 * h1 + b2),
    // Compute the loss
    Step3.3: Compute the **cross_entropy_loss**(*Mean Squared Error (MSE)*)
        between the predicted output $Y\hat{}$ and the target output $Y$.
            MSE = (1/n) * $\sum$( $Y\hat{} - Y$)^2
    //n:batch size
    Step3.4:
    //Update the model parameters W and b using backpropagation and a gradient descent optimizer (e.g., Adam((Adaptive Moment Estimation) optimizer)).
    Step 3.5:grad = compute_gradient(L, W)
    Step 3.6:W = update_weights(W, grad, learning_rate)
    Step 3.7:return $Y\hat{}$ // Its represents candidate keyphrase vector fixed length 512 dimensions.
End
```

```
End
```

3.6 Preserve informative keyphrase

To obtain the most informative Keyphrases using(MMR) method, which combines in a controllable way the concepts of relevance and diversity , in other words, combines keyphrase informativeness with dissimilarity among the selected keyphrases, The MMR method was used by applying Equation 2.10.

The steps of the remove Redundant candidate keyphrase for each, document are illustrated in Algorithm 3.4.

Algorithm 3.4: Maximal Marginal Relevance(MMR)

```
Input:
Document text, Pre-trained sentence embedding model, Candidate keyphrases
embedding vector, Maximum number of keyphrases to select (No.of.k), λ=0.5value
(λ) for the trade-off between relevance and diversity
Output:
Selected key phrases based on the MMR algorithm
Begin
    Step1:Encode the full document text into a document embedding using the
          pre-trained sentence embedding model.
    Step2:Calculate the similarity score between the document embedding and
          the embedding of each candidate keyphrase.
    Step3:Initialize an empty set A for selected key phrases.
    Step4:Initialize an empty set B for the remaining candidate keyphrases.
    Step5:Add the candidate keyphrase with the highest similarity score to the
          set A.
```

```
Step6:Remove the selected keyphrase from the set of candidate keyphrases.
Step7:While the number of selected keyphrases is less than No.of.k:
    Step7.1:For each remaining candidate keyphrase b in the set B:
        Step7.1.1: Calculate the maximum marginal relevance (MMR) score for b as:
MMR(b) = λ * similarity(b, document) - (1 - λ) * max(similarity(b, a)) for a in A// apply Equation 2.10
    Step7.2: Select the candidate keyphrase b with the highest MMR score.
    Step7.3: Add b to set A.
    Step7.4: Remove b from set B.
Step8: Return the set of selected keyphrases A.
End
```

In Figure 3.6 this step is summarized

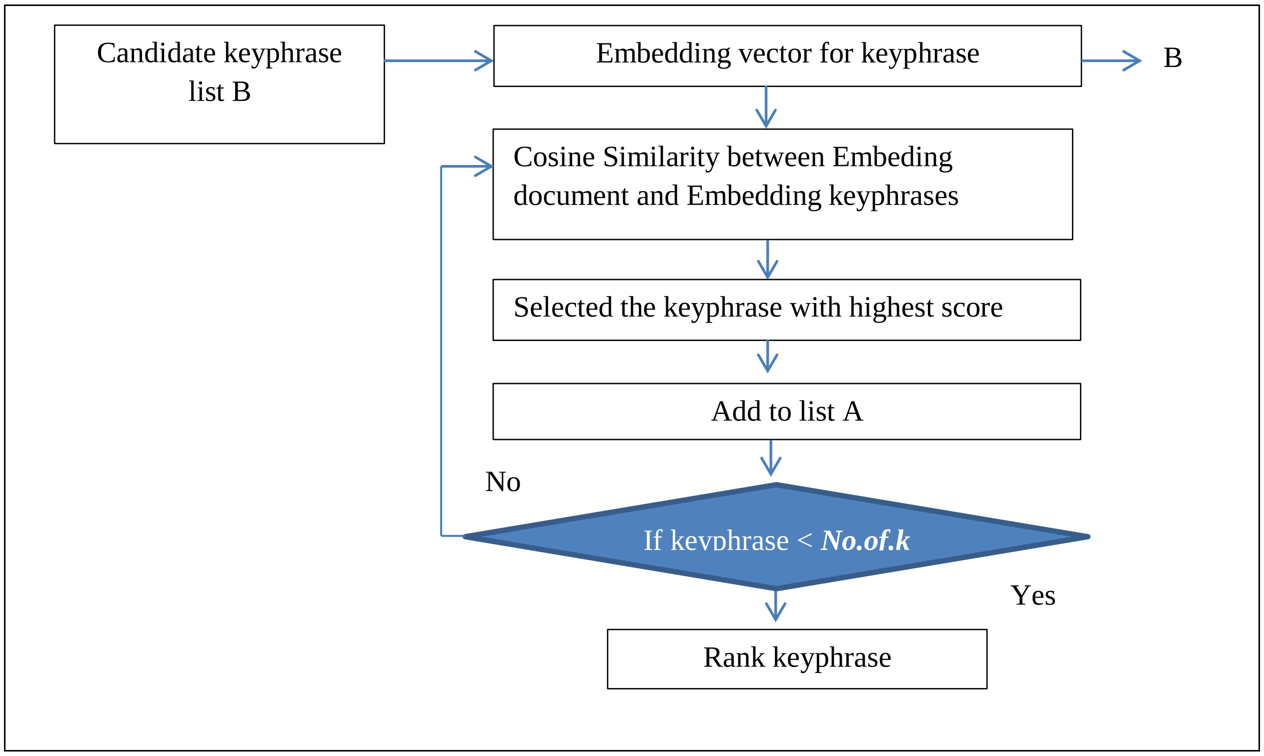

Candidate keyphrase list B
Embedding vector for keyphrase
B
Cosine Similarity between Embeding document and Embedding keyphrases
Selected the keyphrase with highest score
Add to list A
No
If keyphrase < No.of.k
Yes
Rank keyphrase

Figure 3.6:Maximal Marginal Relevance Method(MMR)

3.7 Approaches indexing and ranking

for indexing and ranking, two approaches are considered: the first method creates an embedding document vector by normalization embedding vector rank keyphrases for this document then cosine similarity with the embedding query vector. Then build the second method for enhancement above method, through constructing an inverted index to compute and rank the most similar ones with query embedding vector and finally score the most documents relevant to the query vector.

3.7.1 Document Average Embedding

This method creates an embedding document vector by normalization embedding vector rank keyphrases for this document then cosine similarity with embedding queries vectors. then retrieve top10 ranking documents after that compare retrieved documents with the judgment file finally apply the evaluation metric.

1. **Normalization selected keyphrases vector**

to obtain document vector by calculating average vector illustrated in Figure 3.1.

$$\text{Avg}=(\sum_{i=1}^{no.of.k} keyphrase(i))/no.of.k \text{ .... } (3.1)$$

Example

d=[Vk1,Vk2,..........,Vkn] //d: represent a document, VK: represent vector keyphrase.

Vk1=[0.5,0.7,0.4,.........,0.3] //length vector 512 dimension

Vk2=[0.2,0.2,0.3,.........,0.1]

.

.

Vkn=[0.8,0.3,0.1,..........,0.5]

Sum=[1.5,1.2,0.8,..........,0.9]/$no.of.k$ // $no.of.k$=3

Avg.=[0.5,0.4,0.266,......,0.3]

The values of (Avg.) vector represent document embedding vector.

The steps of create document embedding vector are illustrated in Algorithm 3.5.

Algorithm 3.5: Create document embedding vector

```
Input: list of selected keyphrase
Output: dictionary of average document embedding vector
Begin
 Step1: For each document
   Begin
      For all list of selected keyphrase
       Begin
          Create keyphrase vector embedding VK
        End
        Calculate Averaging for each document
        Vector=($\sum_{i=1}^{no.of.Vk} Vector\ keyphrases(i))/no.of.Vk$
       End
   Step2:Return dictionary of document embedding vectors [name of the document, average document embedding vector]
End
```

2. Cosine similarity

The cosine method was used by applying Equation 2.12 to find the similarity between the vector of query and vectors of documents. Algorithm 3.6 illustrates the main steps of cosine similarity.

Algorithm 3.6: Calculating similarity using the cosine method(document average embedding vector)

```
Input: queries of the data, dictionary document of vector
Output: Dictionary of cosine scores [key, value] contain documents that are more similar to the query, where key: represents document name, value: represents similarity degree.
Begin
  Step 1:For each q in queries  //q :represents query
    Begin
    Step1.1:preprocessing  q by  using algorithm 3.1
    Step1.2:For each d in dictionary document of vector //d: represents a document
        create q vector V(q) by using  algorithm 3.3//V(q) :vector query,V(d):vector document
          For each pair (d, q)  // q: represents a vector of the query
              Calcualte cosine similarity
                    cosine= cosine similarity(V(d),V(q) )
          End
      End
End
Step2: Return sorted in a descending  Dictionary of cosine scores [document number, similarity score]    //where represent the top N retrieved documents.
End
```

3. Retrieval Evaluation

The mean average precision (MAP) assessment measure in Equation 2.15 was used to calculate the efficiency of the retrieval of this system after the ranked documents were retrieved using the cosine method. Enhancement of this results in a method1 through applying method2 as shown below.

3.7.2 Build Inverted Index

The inverted index consists of a mapping between terms and the documents in which they appear. In this thesis, the inverted index consists of a mapping between keyphrases and the posting list of the documents. It allows for quick lookups of documents that contain specific keyphrases, making it a crucial part of search engines and text-based retrieval systems.

1. **Dictionary Key phrase extraction**

Keyphrase extraction is to identify the most salient and representative phrases that best represent the content of a document. These key phrases can help in organizing and categorizing large amounts of textual data.

2. **Posting list of documents identifiers**

A posting list represents a specific term and contains a list of document identifiers (or other unique identifiers) where the term appears. Each entry in the posting list typically includes additional information, such as the frequency or position of the term within the document. Table 3.8 illustrates an example of a posting list.

Table 3.8: An example of a posting list

Term	Posting list
algorithm	[Doc1, Doc2, Doc4, Doc5, Doc6, Doc7, Doc8]
database	[Doc2, Doc3, Doc5, Doc8, Doc9, Doc10]

In this example, the term "algorithm" appears in documents Doc1, Doc2, Doc4, Doc5, Doc6, Doc7, and Doc8. The term "database" appears in documents Doc2, Doc3, Doc5, Doc8, Doc9, and Doc10.

3. Cosine similarity

Appling the similarity between the vector of the query and vectors for each key phrase in a document in an inverted index. Algorithm 3.7 illustrates the main steps of cosine similarity.

Algorithm 3.7: Calculating similarity using the cosine method(inverted index)

```
Input: queries of the data ,Inverted index
Output: Dictionary of cosine scores [key, value] contain documents that are more similar to the query, where key: represents document name, value: represents similarity degree.
Begin
  Step 1:For each q in queries  //q :represents query
     Step1.1:preprocessing  q by  using algorithm 3.1
     Step1.2:Create q vector V(q)   using  algorithm 3.3
     Step1.3:For each d in Posting list //d: represents a document
             For each k in d     //k: keyphrases appear in document
               For each pair (q, k)
                  //Calcualte cosine similarity
```

```
                    Cosine_Keyphrases=cosine similarity(V(q),V(k))
                 End
               End
               //Normalization Cosine Key phrases
              For cos in Cosine_Keyphrases
                     doc_cosine=$(\sum_{i=1}^{no.of.k} \cos_i)/no.of.k$
               End
      End
   End
   Step2: Return sorted in a descending  Dictionary of cosine scores  [document
number, similarity score]   //where represent the top N retrieved documents.
End
```

4. Mean average precision

Finally, apply The mean average precision (MAP) assessment measure was used to calculate the efficiency of the retrieval of this system after the ranked documents were retrieved using the cosine method.

Chapter Four: Results and Discussion

4.1 Introduction

This chapter describes the experimental outcomes for the suggested system for each point as discussed in chapter three. To demonstrate this system's success, we'll explain a case study here. We chose several values of parameter N(number of retrieved documents) to get the best result.

4.2 System Requirement

Hardware: Processor Intel i5, RAM 16GB, Freq. 2.40GHz.

Operating System: Windows10 Education,64-bit.

Programming language: Python in visual code: the system was implemented by Python 3.10.11

4.3 Case study

The document with (ID en.15.110.376.2009.3.24) and the query with (ID148) are chosen as an example of an input to implement the system.

4.3.1 Preprocessing Stage

The preprocessing contains tokenization, lower case, taking one-word consideration less important words which are called stop words and deleting them. Applying stemming, as shown below. The query with (ID 148) is displayed in Figure 4.1.

```
<top lang='en'>
<num>148</num>
<title>Popularity of social networking sites</title>
<desc>Worldwide popularity of social networking sites</desc>
<narr>The popularity of the social networking sites the world over and the reasons for their
gathering such large fan following.</narr>
</top>
```

Figure 4.1:Query Sample

1. **Lower case: Table 4.1 illustrates the results after** transforming the query to lowercase.

Table 4.1:Term Lower casing

Before lowercase	Popularity of social networking sites
After lowercase	popularity of social networking sites

2. **Word Tokenization:** The outcomes of separating query texts into a list of terms are illustrated in Table 4.2

Table 4.2:Word Tokenization

Before Tokenization	popularity of social networking sites
After Tokenization	['popularity', 'of', 'social', 'networking', 'sites']

3. **Removing Stop Words:**

Stop words are unimportant words, which have to repeat frequently. Table 4.3 illustrates the query after removing stop words.

Table 4.3: Removing Stop Word

Before remove	['popularity', 'of', 'social', 'networking', 'sites']
After remove	['popularity', 'social', 'networking', 'sites']

4. **Stemming Stage**:

This stage means returning words of the query to their root forms. Table 4.4 illustrates the results after applying the stemming of the query.

Table 4.4: Stemming Stage

Before stemming	['popularity', 'social', 'networking', 'sites']
After remove	popular social network site

Then applying steps of preprocessing on this document and the outcome are given in Figure 4.3. Figure 4.2 displays the document with ID (en.15.110.376.2009.3.24)

<DOC>
<DOCNO>en.15.110.376.2009.3.24</DOCNO>
<TITLE> Netflix integrating movie ratings with Facebook </TITLE>
<TEXT>
LOS ANGELES, Mar 24 (bdnews24.com/Reuters) - Netflix Inc is the latest media company to integrate with social networking website Facebook, whose huge community of young, tech-savvy users could help drive growth of the online DVD rental service's subscriber base. Starting on Tuesday, Netflix users can use Facebook Connect -- software that links individual Facebook pages to third-party Web sites -- to share their ratings of Netflix rentals with their Facebook friends, the companies said in a statement. "Intuitively, the folks streaming (Netflix movies) on the laptop tend to be the under-25 crowd," Netflix spokesman Steve Swasey said. "You could make the leap that it is the more tech-savvy ... the early adopter crowd

... but Facebook is becoming more mainstream." Movie ratings will appear on Netflix subscribers' Facebook pages if they opt into the program, and will link back to the correlating movie page at Netflix.com, the companies said. The tie-up puts Netflix's brand and its fast-growing online streaming service -- a major growth driver for its subscriber base -- in front of more than 175 million active Facebook users through what is essentially a marketing channel. Facebook Connect, launched last year, was expected to transform the social network from a private site where activity occurs entirely within a "wall garden" to a Web-wide phenomenon where software makers, with user permission, can tap member data for use on their sites. Among others, early partners included CBS, Disney-ABC, Discovery.com and Hulu, jointly owned by News Corp and NBC Universal
</TEXT>
</DOC>

Figure 4.2: Document sample

netflix integr movi rate facebook angel mar bdnew com reuter netflix inc latest media compani integr social network websit facebook whose huge commun young tech savvi user could help drive growth onlin dvd rental servic subscrib base start tuesday netflix user use facebook connect softwar link individu facebook page third parti web site share rate netflix rental facebook friend compani said statement intuit folk stream netflix movi laptop tend crowd netflix spokesman steve swasey said could make leap tech savvi earli adopt crowd facebook becom mainstream movi rate appear netflix subscrib facebook page opt program link back correl movi page netflix com compani said tie put netflix brand fast grow onlin stream servic major growth driver subscrib base front million activ facebook user essenti market channel facebook connect launch last year expect transform social network privat site activ occur entir within wall garden web wide phenomenon softwar maker user

```
permiss tap member data use site among earli partner includ disney abc discoveri com hulu jointli news corp nbc univers
```

Figure 4.3: Document_Preprocessing Sample

4.3.2 Extract candidate keyphrase

Extract candidate key phrases by extracting noun phrases based on Regular expression depending on a Part-of-Speech (POS) tagger.

1. Part-of-Speech (POS) Tagger

Applying the part of speech tagger(pos) **by using standford corenlp** for this document **illustrated in Figure 4.4**

```
[[('netflix', 'NN'), ('integr', 'NN'), ('movi', 'NN'), ('rate', 'NN'), ('facebook', 'NN'), ('angel', 'NN'), ('mar', 'FW'), ('bdnew', 'FW'), ('com', 'NN'), ('reuter', 'NN'), ('netflix', 'NN'), ('inc', 'NN'), ('latest', 'JJS')], [('media', 'NNS'), ('compani', 'NNS'), ('integr', 'VBP'), ('social', 'JJ'), ('network', 'NN'), ('websit', 'NN'), ('facebook', 'NN'), ('whose', 'WP$'), ('huge', 'JJ'), ('commun', 'JJ'), ('young', 'JJ'), ('tech', 'NN'), ('savvi', 'NN'), ('user', 'NN'), ('could', 'MD'), ('help', 'VB'), ('drive', 'VB'), ('growth', 'NN'), ('onlin', 'NN'), ('dvd', 'NN'), ('rental', 'NN'), ('servic', 'JJ'), ('subscrib', 'NN'), ('base', 'NN'), ('start', 'NN')], [('tuesday', 'NNP'), ('netflix', 'NNP'), ('user', 'NN'), ('use', 'NN'), ('facebook', 'NN'), ('connect', 'VB'), ('softwar', 'NN'), ('link', 'NN'), ('individu', 'NN'), ('facebook', 'NN'), ('page', 'NN')], [('third', 'JJ'), ('parti', 'NN'), ('web', 'NN'), ('site', 'NN'), ('share', 'NN'), ('rate', 'NN'), ('netflix', 'NN'), ('rental', 'NN'), ('facebook', 'NN'), ('friend', 'NN'), ('compani', 'NNS'), ('said', 'VBD'), ('statement', 'NN')], [('intuit', 'JJ'), ('folk', 'NN'), ('stream', 'NN'), ('netflix', 'NN'), ('movi', 'NN'), ('laptop', 'NN'), ('tend', 'VBP'), ('crowd', 'NN'), ('netflix', 'NN'), ('spokesman', 'NN'), ('steve', 'NNP'), ('swasey', 'NNP')], [('said', 'VBD'), ('could', 'MD'), ('make', 'VB'), ('leap', 'NN'), ('tech', 'NN'), ('savvi', 'NN'), ('earli', 'NNS'), ('adopt', 'VBP'), ('crowd',
'NN'), ('facebook', 'NN'), ('becom', 'NN'), ('mainstream', 'NN')], [('movi', 'NN'),
('rate', 'NN'), ('appear', 'VBP'), ('netflix', 'NN'), ('subscrib', 'NN'), ('facebook', 'NN'),
('page', 'NN'), ('opt', 'VBP'), ('program', 'NN'), ('link', 'NN'), ('back', 'RB'), ('correl',
'JJ'), ('movi', 'NN')], [('page', 'NN'), ('netflix', 'NN'), ('com', 'NN'), ('compani',
'NNS'), ('said', 'VBD'), ('tie', 'NN'), ('put', 'VBN'), ('netflix', 'NN'), ('brand', 'NN'),
('fast', 'JJ'), ('grow', 'NN'), ('onlin', 'NN'), ('stream', 'NN'), ('servic', 'NN')], [('major',
'JJ'), ('growth', 'NN'), ('driver', 'NN'), ('subscrib', 'NN'), ('base', 'NN'), ('front', 'NN'),
('million', 'CD'), ('activ', 'NN'), ('facebook', 'NN'), ('user', 'NN'), ('essenti', 'NN'),
('market', 'NN')], [('channel', 'NN'), ('facebook', 'NN'), ('connect', 'VB'), ('launch',
'NN'), ('last', 'JJ'), ('year', 'NN'), ('expect', 'VBP'), ('transform', 'VB'), ('social', 'JJ'),
('network', 'NN'), ('privat', 'NN')], [('site', 'NN'), ('activ', 'NN'), ('occur', 'VBP'),
('entir', 'NN'), ('within', 'IN'), ('wall', 'NN'), ('garden', 'NN'), ('web', 'NN'), ('wide',
'JJ'), ('phenomenon', 'NN'), ('softwar', 'NN'), ('maker', 'NN'), ('user', 'NN')],
[('permiss', 'JJ'), ('tap', 'NN'), ('member', 'NN'), ('data', 'NNS'), ('use', 'VBP'), ('site',
'NN'), ('among', 'IN'), ('earli', 'JJ'), ('partner', 'NN'), ('includ', 'NN'), ('disney', 'NN'),
('abc', 'NN'), ('discoveri', 'NN'), ('com', 'NN'), ('hulu', 'NN'), ('jointli', 'NN'), ('news',
'NN'), ('corp', 'NN'), ('nbc', 'NN'), ('univers', 'NNS')]]
```

Figure 4.4:Part of speech tagger document

2. **Apply regular expression**

Extract candidate keyphrase(noun phrase) when grammer expression equal to "{ <JJ>*<NN.+>}"illustrated in Figure 4.5 .

```
['site activ', 'site', 'leap tech savvi earli', 'com reuter netflix inc', 'movi rate',
'channel facebook', 'netflix subscrib facebook page', 'tuesday netflix user use
facebook', 'correl movi', 'crowd netflix spokesman steve swasey', 'page netflix
```

com compani', 'social network websit facebook', 'statement', 'program link', 'media compani', 'launch last year', 'activ facebook user essenti market', 'softwar link individu facebook page', 'tie', 'crowd facebook becom mainstream', 'permiss tap member data', 'entir', 'social network privat']

Figure 4.5:Extract candidate keyphrase document

4.3.3 Create Embedding Vector

The vector embedding depends on USE(Deep averaging network) model to create vector 512 fixed-dimensional for each candidate keyphrase document and query.

1. **Candidate key phrase vector:** representation of one of the candidates as shown in Table 4.5

Table 4.5: Candidates Embedding Vector

Candidate keyphrase	Embedding Vector
social network privat	[-0.03815525 0.05244004 -0.0307271 0.01023942 0.03599579 0.08676676 -0.0424294 ..... -0.01971244 -0.01465436 -0.05703105 -0.03785515 -0.02363321 0.05536275 -0.03363672 -0.04613208]]

2. **Query Embedding:** representation of query as shown in Table 4.6

Table 4.6: Query Embedding Vector

Query	Embedding Vector
popular social network site	[-5.4786284e-02 6.2135819e-02 -3.0206904e-02

	-1.8851789e-02 7.0282589e-03 .... -5.5370387e-03 -4.2373039e-02 2.9577920e-02]]

4.3.4 Preserve informative keyphrase using Maximal Marginal relevance(MMR)

Remove redundant candidate key phrases by using MMR to obtain more informative key phrases to represent the document and the maximum selected keyphrase is 10, illustrated in Figure 4.6.

```
['com reuter netflix inc', 'netflix subscrib facebook page', 'social network websit facebook', 'softwar link individu facebook page', 'social network privat']
```

Figure 4.6:Selected keyphrases

4.3.5 Approaches indexing and ranking

1. Document Average Embedding vector

This approach is the document embedding vector process, illustrated in section 3.8.1. Then build a dictionary document vector. After creating the document embedding vector need to build a dictionary[key, value] containing in key: document , value: vector embedding, this phase using when done similarity between query and document illustrated in Table 4.7

Table 4.7:Dictionary documents

key	Value
' en.15.110.376.2009.3.24'	[[1.11237448e-02 -2.78138872e-02 7.72176031e-03 1.27486065e-02 6.95999945e-03 -1.11439463e-03 1.77162532e-02 .... -1.10439789e-02 -1.67164877e-02]

Then calculate cosine similarity beween document embedding vector and query embedding vector. As shown result in Table 4.8

Table 4.8 Displays the effects of cosine resemblances between query and document.

Vector1	Vector 2	Cosine similarity
Query(148)	Doc.(en.15.110.376.2009.3.24)	0.414271

And the first top 10 documents retrieved and their ranking are shown in Table 4.9.

Table 4.9: Top 10 documents retrieved and their ranking (Document Average Embedding)

ID	Query	Document	Cosine similarity
1	popular social network site	en.15.110.376.2009.3.24	0.414271
2	popular social network site	en.3.333.71.2009.10.28	0.308149
3	popular social network site	en.3.347.413.2010.2.5	0.303438
4	popular social network site	en.15.66.289.2008.5.8	0.294508
5	popular social network site	en.3.349.356.2010.2.13	0.278987
6	popular social network site	en.15.139.371.2009.8.11	0.274527
7	popular social network site	en.3.339.474.2009.12.7	0.273594
8	popular social network site	en.15.181.237.2010.5.19	0.259468
9	popular social network site	en.15.201.453.2007.8.9	0.254755
10	popular social network site	en.2.278.240.2009.8.24	0.235410

Then calculate the precision equal (Count divided by ID) for each of these Top 10 rankings are shown in Table 4.10.

Table 4.10: Top 10 documents retrieved and their ranking and precision (Document Average Embedding)

ID	Query	Document	Relevant or non-relevant	Count	Presision
1	popular social network site	en.15.110.376.2009.3.24	1	1	1.0
2	popular social network site	en.3.333.71.2009.10.28	0	1	0.5
3	popular social network site	en.3.347.413.2010.2.5	1	2	0.666
4	popular social network site	en.15.66.289.2008.5.8	1	3	0.75
5	popular social network site	en.3.349.356.2010.2.13	1	4	0.8
6	popular social network site	en.15.139.371.2009.8.11	1	5	0.833
7	popular social network site	en.3.339.474.2009.12.7	0	5	0.7142
8	popular social network site	en.15.181.237.2010.5.19	0	5	0.625
9	popular social network site	en.15.201.453.2007.8.9	1	6	0.6666
10	popular social network site	en.2.278.240.2009.8.24	1	7	0.7

2. Inverted Index

This approach is proposed when the result mean average precision(MAP) is low in first method, different from Document Average Embedding approach after the selected keyphrase builds an Inverted Index from the keyphrases and posting list are shown in Table 4.11.

Table 4.11: Inverted Index

Dictionary	Posting list
‘servic facebook twitter'	['en.13.1.289.2009.6.2']
'senior vice’	['en.13.1.289.2009.6.2']
'servic april'	['en.13.1.306.2009.6.2','en.15.81.424.2008.8.26']
‘presid microsoft'	['en.13.1.306.2009.6.2']
' commun system'	['en.13.1.306.2009.6.2']
' terrorist act sinc terror'	['en.13.1.306.2009.6.2']
' main threat'	['en.13.1.306.2009.6.2']
' brazilian airlin tam'	['en.13.1.306.2009.6.2','en.15.88.271.2008.10.9']
'next year'	['en.13.1.360.2009.5.30','en.13.10.57.2009.7.29', 'en.13.25.179.2009.11.8','en.13.3.388.2009.6.12', 'en.13.8.152.2009.7.16','en.15.100.203.2008.12.12', 'en.15.100.343.2008.12.15','en.15.100.486.2008.12.15', 'en.15.101.122.2008.12.17','en.15.106.477.2009.1.22', 'en.15.109.444.2009.3.31' 'en.15.111.470.2009.3.18']

Then calculate the cosine similarity between the key phrases vector for this document from the inverted index and query 148, as shown in Table 4.12

Tabel 4.12: Keyphrase Score

Document	Keyphrase	Cosine similarity
en.15.110.376.2009.3.24	'com reuter netflix inc'	0.345
en.15.110.376.2009.3.24	'netflix subscrib facebook page'	0.889
en.15.110.376.2009.3.24	'social network websit facebook'	0.77

en.15.110.376.2009.3.24	'softwar link individu facebook page'	0.432
en.15.110.376.2009.3.24	'social network privat'	0.978

Finally do normalization for the above cosine similarity key phrases to obtain document score using when retrieval ranking documents according to this score as shown in Table 4.13.

Table 4.13 Displays the effects of cosine resemblances between query and document.

Vector1	Vector 2	Cosine similarity
Query(148)	Doc.(en.15.110.376.2009.3.24)	0.6828

The first top 10 documents retrieved and their ranking are shown in Table 4.14.

Table 4.14: Top 10 documents retrieved and their ranking (Inverted Index)

ID	Query	Document	CosineSimilarity
1	popular social network site	en.15.110.376.2009.3.24	0.6828
2	popular social network site	en.3.347.413.2010.2.5	0.611344
3	popular social network site	en.15.139.371.2009.8.11	0.578093
4	popular social network site	en.3.333.71.2009.10.28	0.560988
5	popular social network site	en.3.349.356.2010.2.13	0.549766
6	popular social network site	en.15.66.289.2008.5.8	0.528932
7	popular social network site	en.2.281.347.2009.9.16	0.522130
8	popular social network site	en.15.181.237.2010.5.19	0.510272
9	popular social network site	en.15.250.237.2008.3.13	0.499764

10	popular social network site	en.15.92.355.2008.10.30	0.494203

Then calculate the precision equal (Count divided by ID) for each of these Top 10 rankings are shown in Table 4.15.

Table 4.15: Top 10 documents retrieved and their ranking and precision(Inverted Index)

ID	Query	Document	Relevant or non relevant	Count	Presision
1	popular social network site	en.15.110.376.2009.3.24	1	1	1.0
2	popular social network site	en.3.347.413.2010.2.5	1	2	1.0
3	popular social network site	en.15.139.371.2009.8.11	1	3	1.0
4	popular social network site	en.3.333.71.2009.10.28	0	3	0.75
5	popular social network site	en.3.349.356.2010.2.13	1	4	0.8
6	popular social network site	en.15.66.289.2008.5.8	1	5	0.833
7	popular social network site	en.2.281.347.2009.9.16	1	6	0.85714
8	popular social network site	en.15.181.237.2010.5.19	0	6	0.75
9	popular social network site	en.15.250.237.2008.3.13	1	7	0.77777
10	popular social network site	en.15.92.355.2008.10.30	1	8	0.8

4.4 Evaluation

In this section, we will note the results of the proposed work according to the evaluation criteria for each of the cases that will be addressed.

4.4.1 Average precision for Top N documents(case study)

Query 148 will be taken as an example to illustrate the influence of various parameters on the outcomes, as indicated in Table 4.16 and Figure 4.7

Table 4.16: Result of Average precision using a different value of N

Case study	**Top N documents**		
(Query 148)	AP@10	AP@20	AP@30
Document Average Embedding	0.72548	0.43321	0.3397
Inverted index	0.856791	0.612434066	0.4536911

calculate the Average precision for both approaches from the summation precision query to document then divide by 10 if top @10 when retrieval ranking top 10 or 20 if top @20 or 30 if top @30.

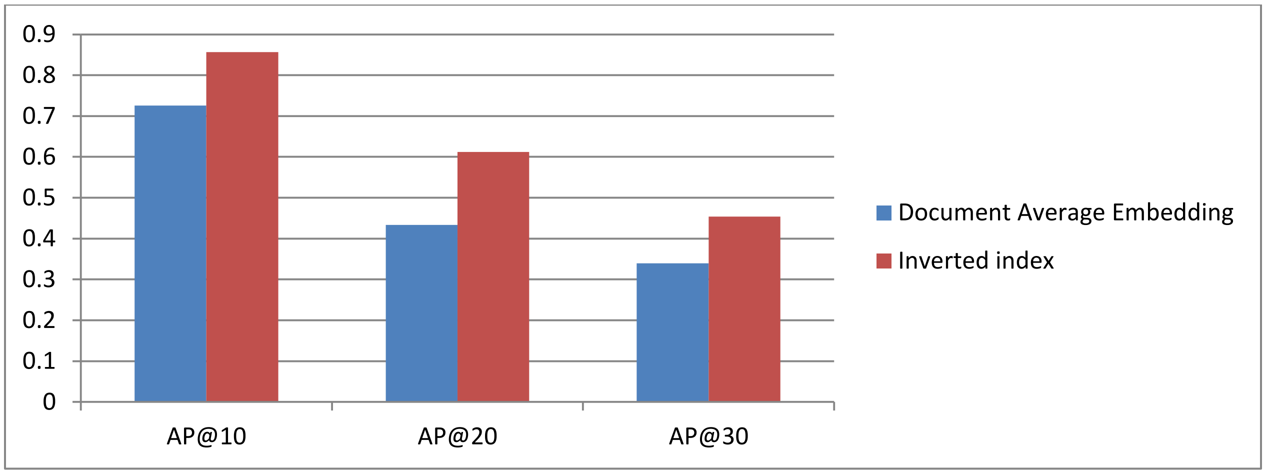


Figure 4.7: Result of Average precision using a different value of N

Rates of performance of the last case are also examined. From the previous table and figure, it can be seen that when parameter N=10 entered the system, the best result for the query was observed.

4.4.2 Average precision for Top N documents(All Queries)

Other queries would also give the system the result by using various values from parameter N. The best performance was demonstrated in Tables 4.17 ,Tabel 4.18 and Figures 4.8, Figures 4.9 for Top 20 queries in FIRE2011 Dataset.

1. Document Average Embedding

Table 4.17: Document average embedding(AP) for the Top 20 Queries

Query No.	**Query**	**TopN documents**		
		AP @10	AP@20	AP@30
126	swine flu vaccin	0.531388	0.71943023	0.49761
127	rare cosmic event	0.8140933	0.82658730	0.2
129	michael jackson untim death	0.909353741	0.91601	0.859325
130	price hike petroleum product	0.5111	0.76666667	0.566666
131	abduct murder journalist	0.90634709	0.73756025	0.707103
132	barack obama victo	0.6359744	0.233144	0.25
133	tortur abu ghraib prison	0.40764	0.57777777	0.590327
134	ban slap simi	0.54333	0.5433	0.399801
136	piraci world entertain	0	0.342	0.4
137	death ltte head	0.414920635	0.33047	0.545833
139	vanquish somali pirat	0.97334163	0.84494	0.676468
140	search life water space	0.720634102	0.9737516	0.488968
141	birth clone human babi	0.303333	0.5	0.42

142	illeg fell tree	0	0.243	0.2
143	tata car nano	0.483333333	0.653962	0.5294
144	bhopal tragedy	0.4	0.3	0.4833
145	assassin benazir bhutto	0.814286	0.67654287	0.687817
146	ram janmabhoomi verdict	0.2675	0.9	0.8
147	cyber crime india	0.7	0.7	0.551700
148	popular social network site	0.72548	0.43321	0.3397

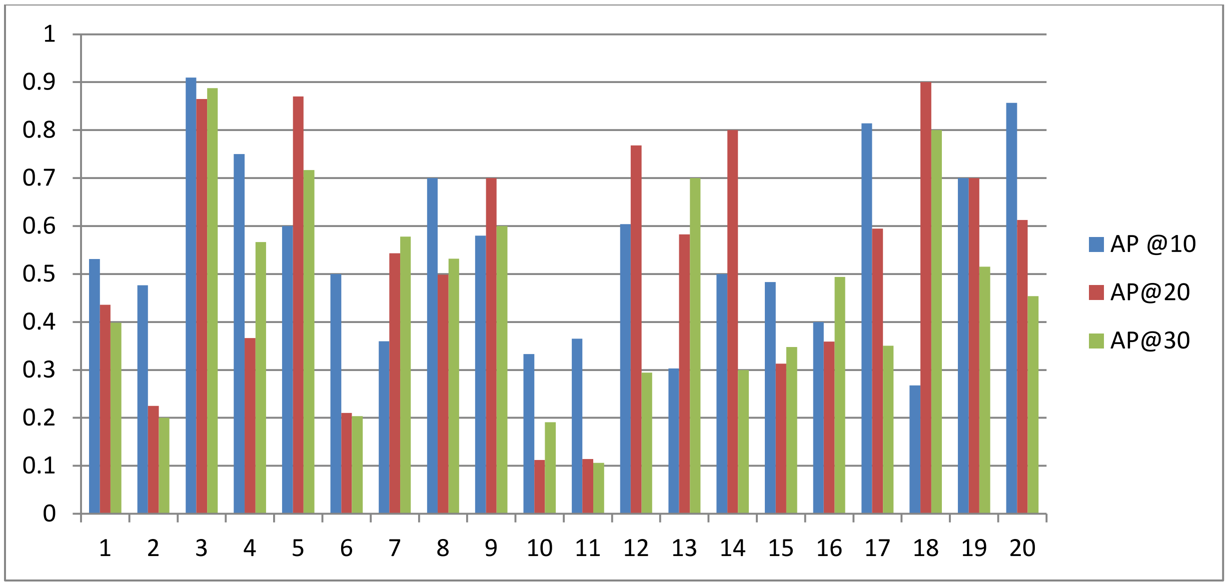


Figure 4.8: AP for 20 Queries(Document Average Embedding)

2. **Inverted Index**

Table 4.18: Inverted index(AP) for the Top 20 Queries

Query No.	Query	TopN documents		
		AP @10	AP@20	AP@30
126	swine flu vaccin	0.531388	0.43616567	0.398708
127	rare cosmic event	0.476666667	0.225	0.2

129	michael jackson untim death	0.909353741	0.86460470	0.887885
130	price hike petroleum product	0.75	0.36666666	0.566666
131	abduct murder journalist	0.6	0.87022816	0.716729
132	barack obama victo	0.5	0.21010634	0.203769
133	tortur abu ghraib prison	0.35952381	0.543	0.577777
134	ban slap simi	0.7	0.49884559	0.531746
136	piraci world entertain	0.58	0.7	0.6
137	death ltte head	0.333333333	0.11212121	0.190909
139	vanquish somali pirat	0.365079365	0.11392354	0.105921
140	search life water space	0.604166667	0.76834968	0.294155
141	birth clone human babi	0.303333	0.58295454	0.7
142	illeg fell tree	0	0.8	0.3
143	tata car nano	0.483333333	0.31319465	0.347756
144	bhopal tragedy	0.4	0.35920803	0.493589
145	assassin benazir bhutto	0.814286	0.59448051	0.350297
146	ram janmabhoomi verdict	0.2675	0.9	0.8
147	cyber crime india	0.7	0.7	0.515178
148	popular social network site	0.856791	0.61243406	0.453691

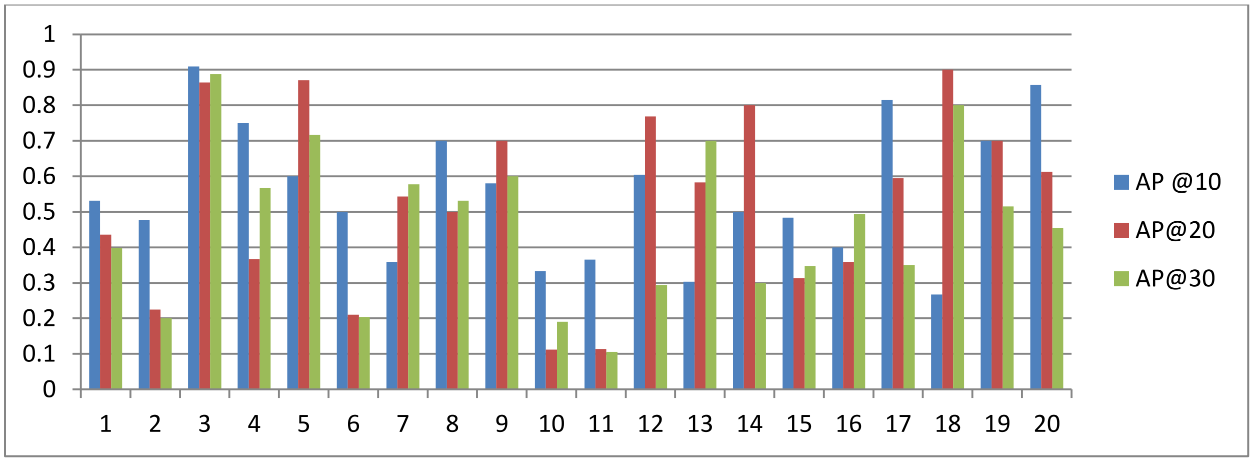


Figure 4.9: AP for 20 Queries(inverted index)

In the two tables above, it was noted that the implementation of the proposed system did not achieve an improvement in retrieval for some inquiries, because the evaluation depends on comparing the retrieved documents with the judgment file, as well as on the number of documents retrieved. For example, query 136 did not achieve an improvement in the system by relying on the document average embedding method until 20 also query 142 did not achieve an improvement in the system by relying on the average embedding method until 30 was retrieved document for that the fewer number of documents are not relevant to the query according to their evaluation within the judgment file, while in the Inverted Index method an evaluation of the query was obtained after retrieving 20 documents.

4.5 Evaluation of Retrieval

4.5.1 Keyphrase

1. Visualization Embedding Vector

We compared the key phrases through visualization embedding vector to identify with this research [14] to note the efficiency of the work of our proposed system in terms of extracting the keyphrases, illustrated in Figure 4.10, Figure 4.11, Figure 4.12, and Figure 4.13. in the figures below, we first transform the candidate keyphrase vector length is 512D into 2D points (x,y) by using the tool Principal component analysis (PCA) is a technique that transforms high-dimension data into lower-dimensions while retaining as much information as possible. PCA is extremely useful when working with data sets that have a lot of features. The figures below indicate that the candidate keyphrases appear very close to each other, as a result of using efficient inclusion that depends on the context. Hence, we conclude that the proximity of the phrases indicates the existence of relationships between them.

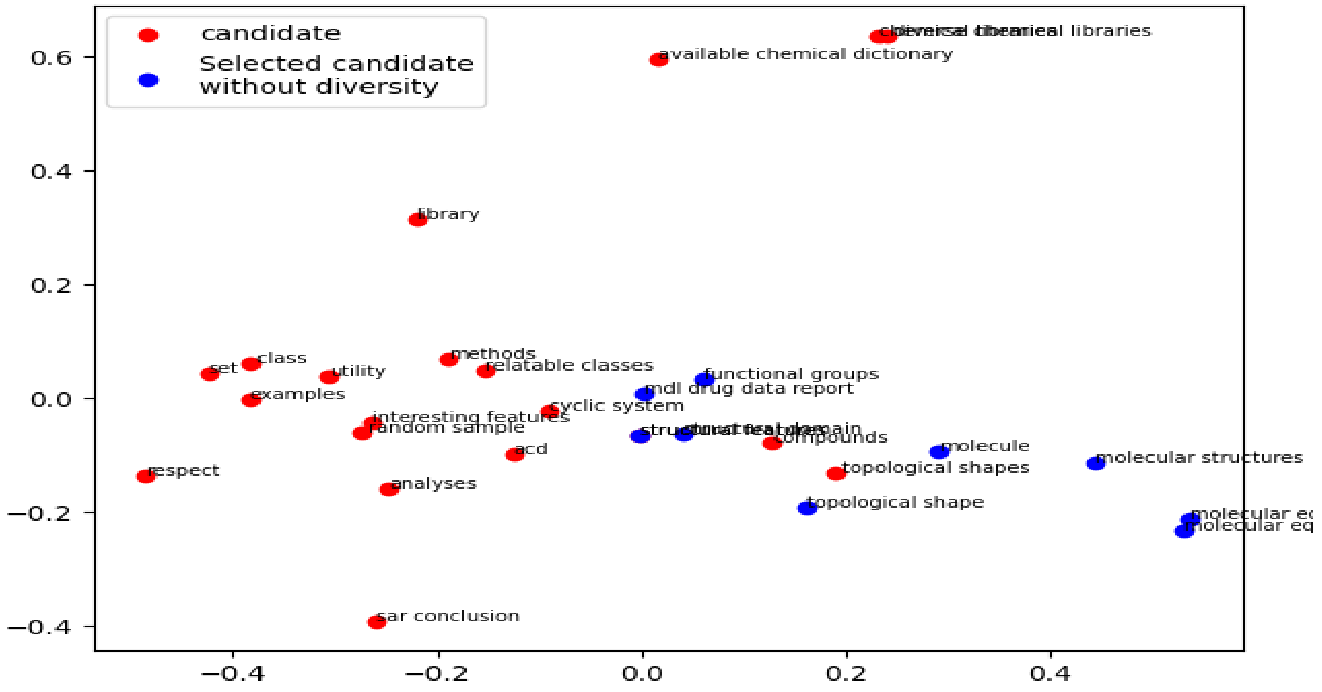


Figure 4.10: Keyphrase without diversity in proposed System

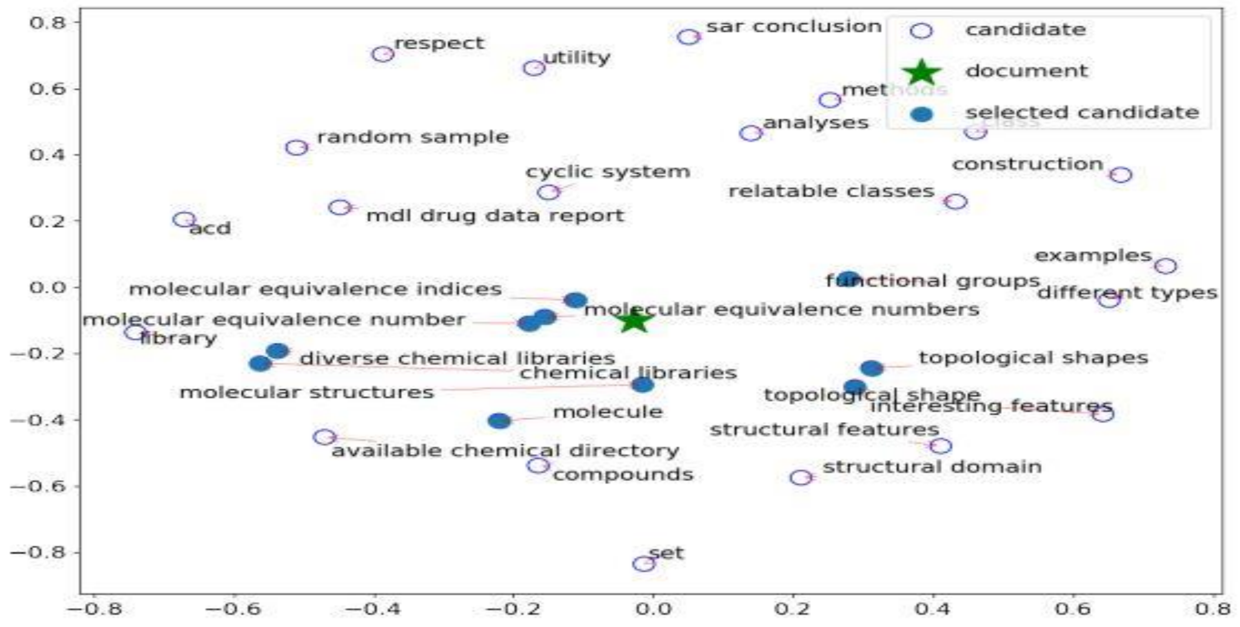


Figure 4.11: Keyphrase without diversity in Kamil Bennani[14].

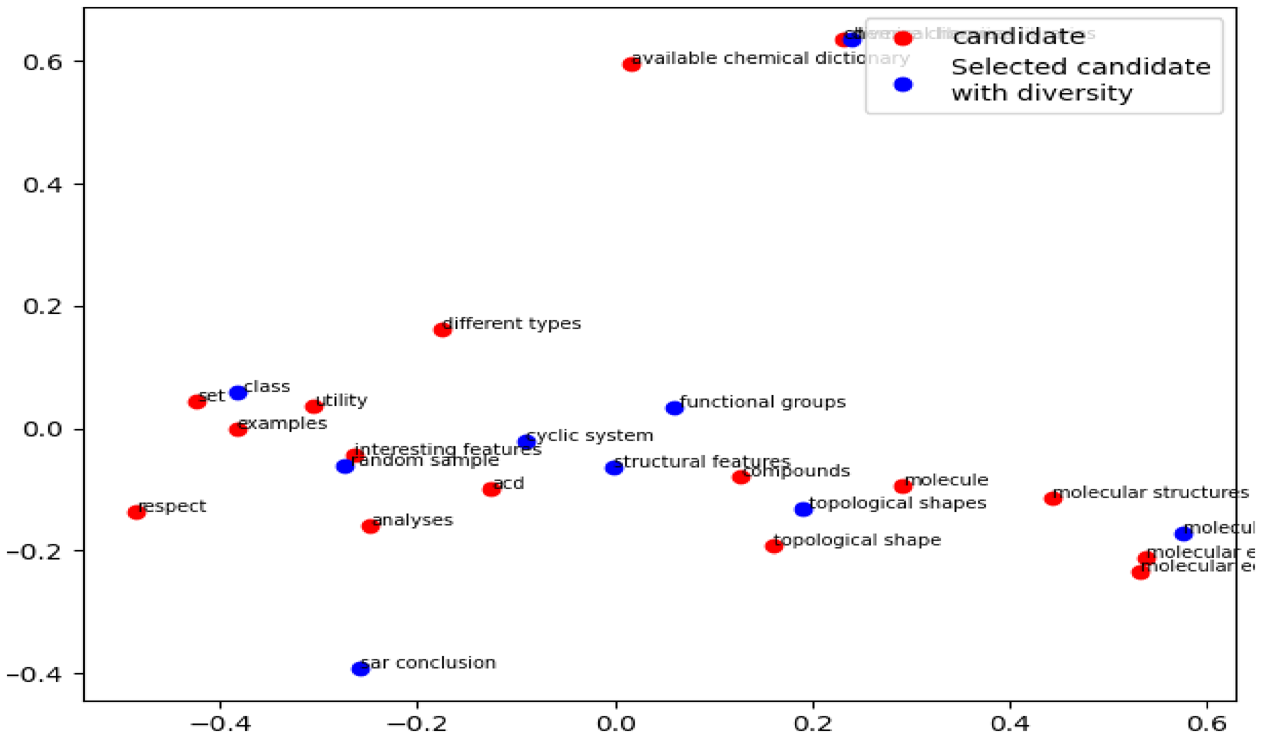


Figure 4.12: Keyphrase with diversity in proposed System.

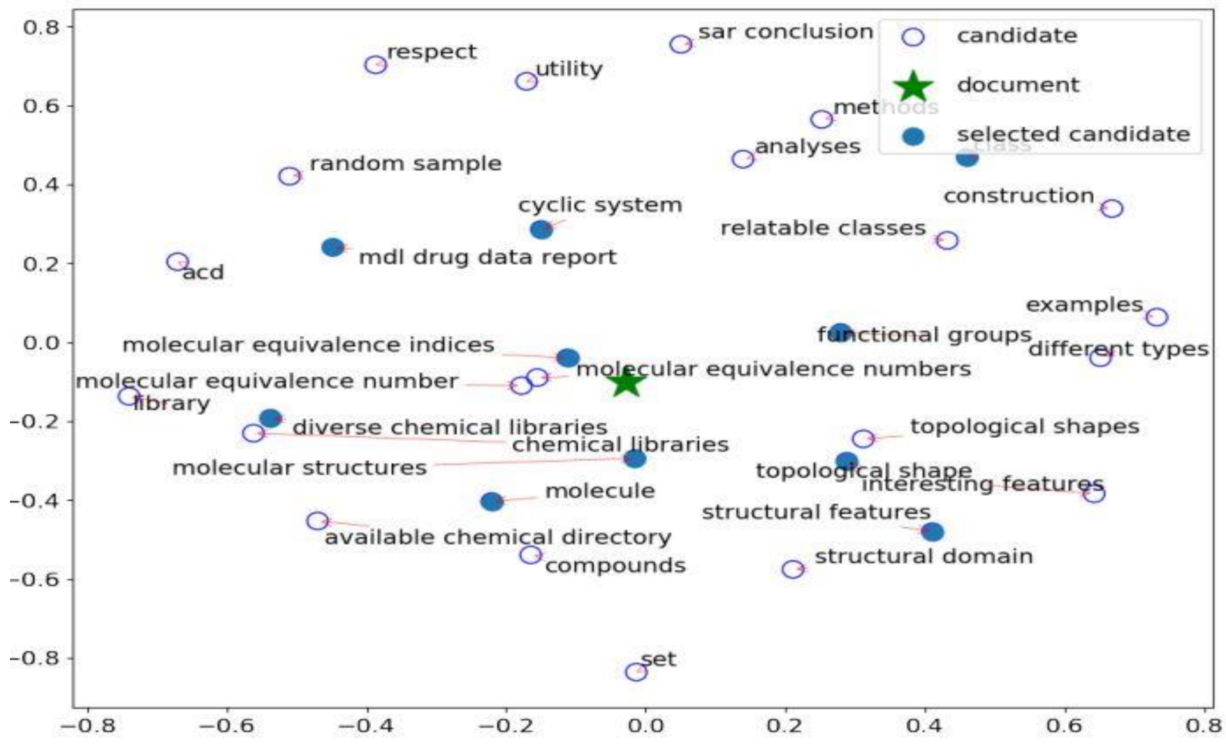


Figure 4.13: Keyphrase with diversity in Kamil Bennani[14].

2. Compare with dataset Inspec

illustrated in Table 4.19. Columns are: number of documents; the average number of tokens per document; the average number of unique candidates per document; the total number of unique keyphrases; average number of unique keyphrases per document; percentage of keyphrases not present in the documents; percentage of keyphrases not present in the candidates; percentage of keyphrases present in the document, but not in the candidates.

Table 4.19:Compare with dataset Inspec

Dataset	**Inspec[14]**	**My proposed**
Document	500	500
Average token	134.63	142.14
Average candidate	26.39	17.06
keyphrases	4903	3933
Average keyphrase	9.81	9.35
Missing keyphrase in document	21.52%	35.24%
Missing due to candidate	18.34%	15.81%

As illustrated in Figure 4.14 difference between the proposed key phrase system and the golden keyphrase Top 20 documents.

As there is a good percentage that we have reached through this comparison with regard to the key phrases, the percentage of similarity is not 100%, but there is a lot of similarity.

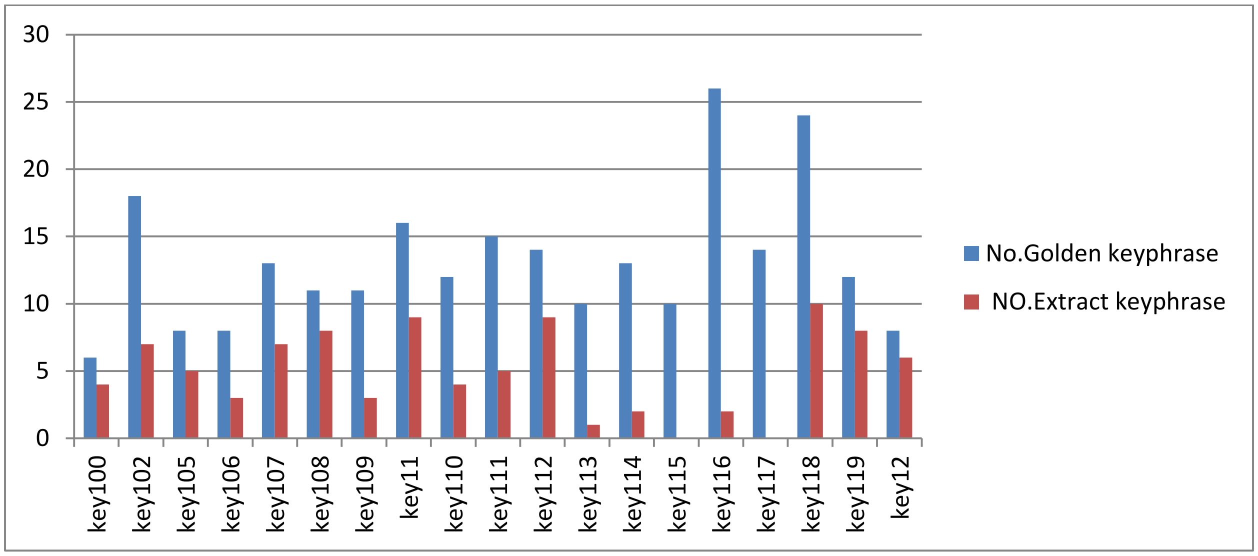


Figure 4.14: Different Keyphrases in Inspec dataset and proposed system .

4.5.2 Document Ranking

Our query sentence-level with embedding results demonstrate that it can increase MAP compared to traditional sentence-level results QE. The results indicate that sentence semantic similarity with embedding is stronger than traditional sentence semantic similarity. Table 4.20 and Figure 4.15 provide a contrast between the MAP different document top N in the system's ranking retrieval.

Table 4.20: Comparison MAP between baseline and approach with embedding using the various values of N

Top N documents	**MAP**		
	Baseline [18]	Semantic similarity with embedding (inverted index)	Semantic similarity with embedding (Document average embedding)
AP @10	0.61	**0.6283**	0.5589172
AP@20	**0.2933**	**0.545**	**0.487**

AP@30	**0.434**	**0.58908**	0.5140644

In the above table, the results of the mean average precision (MAP) scale are shown for all queries within the Dataset.

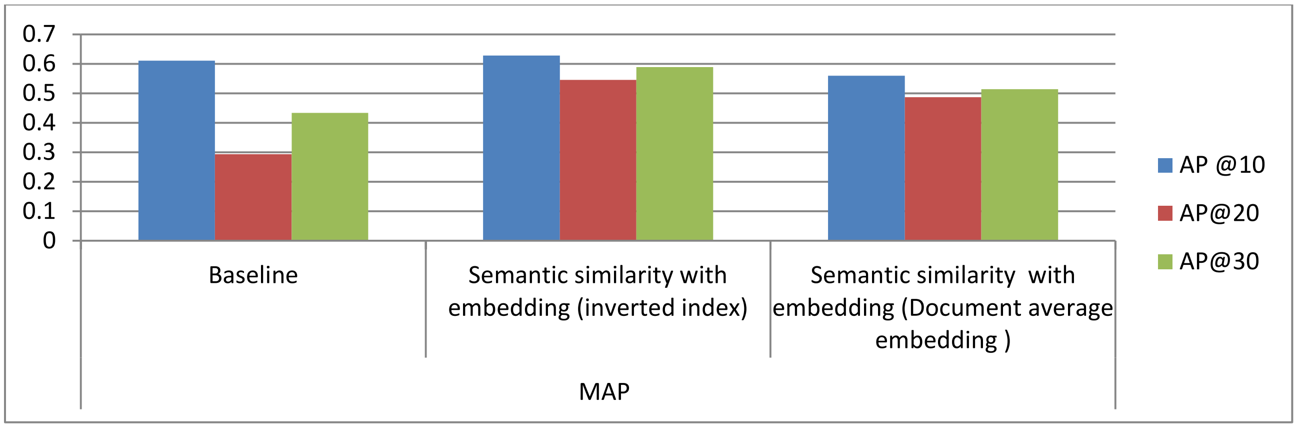


Figure 4.15:Comparsion MAP between baseline and approache with embedding using the various values of N

Our findings were compared to the findings from research[18] which used FIRE2011 as a dataset and semantic similarity between words, as shown in Table 4.21.

Table 4.21: Comparison of Proposed System [18]

Fire2011	**Top 10 document** **MAP**	**Top 20 document** **MAP**
Proposed system (inverted index)	0.6283	0.545
Proposed system (Document average embedding)	0.5589172	**0.487**
[18]	0.61	0.2933

Chapter Five: Conclusions and Future Works

5.1 Conclusions

- ✓ Document keywords and keyphrases enable faster and more accurate search in large text collections ,serve as condensed document summaries
- ✓ embedding the query and keyphrase into the same continuous vector space. This opens the way to computing semantic relatedness among text fragments by using the induced similarity measures in that feature space
- ✓ AKE is crucial to many NLP and information retrieval tasks, including document summarizing, classification, and article recommendation, in this work we consider enhancing document ranking-based information retrieval.
- ✓ Implementing an unsupervised keyphrase extraction method to rank the candidate key phrases according to their importance in the analyzed text.
- ✓ To keep key phrases that have informativeness property we return the N candidate phrases closest to the document embedding by using maximal marginal relevance, since if we extract a fixed number of top keyphrases, redundancy hinders the diversification of the extracted keyphrases.
- ✓ The result for Documents ranking by ***Inverted index*** best than documents ranking by ***Document Average Embedding*** .

5.2 Future Works

Based on the results of this .thesis some suggestions for future work .includes:

1. Applying the proposed system to. other standard collections and topic .sets e.g. the TREC.

2. Applying also Query expansion to improve query with the found keyphrase.

3. Applying the new method to deal with keyphrases not only extraction, use method generation keyphrases.

4. Applying the system to Arabic-language document

References

[1] L. Liu and M. T. Özsu, "Encyclopedia of Database Systems Springer New York." p. 4964, 2018. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/978-1-4614-8265-9_181

[2] S. Ibrihich, A. Oussous, O. Ibrihich, and M. Esghir, "A Review on recent research in information retrieval," *Procedia Computer Science*, vol. 201, no. C. pp. 777–782, 2022. doi: 10.1016/j.procs.2022.03.106.

[3] R. Kumar and S. C. Sharma, "Information retrieval system: An overview, issues, and challenges," *Int. J. Technol. Diffus.*, vol. 9, no. 1, pp. 1–10, 2018.

[4] W. B. Croft, D. Metzler, and T. Strohman, *Search engines: Information retrieval in practice*, vol. 520. Addison-Wesley Reading, 2010.

[5] P.-Y. Yuan, A.-M. Du, and C. Wang, "Using Word2vec to match knowledge points and test questions: a case study," in *2020 IEEE 2nd International Conference on Computer Science and Educational Informatization (CSEI)*, 2020, pp. 272–276.

[6] F. Almeida and G. Xexéo, "Word embeddings: A survey," *arXiv Prepr. arXiv1901.09069*, 2019.

[7] E. Yan and Y. Zhu, "Tracking word semantic change in biomedical literature," *Int. J. Med. Inform.*, vol. 109, pp. 76–86, 2018.

[8] J. Brendl, "Keyword Based Document Retrieval via Document Embeddings." Informatics Institute, 2018.

[9] J. R. Asl and J. M. Banda, “Gleake: Global and local embedding automatic keyphrase extraction,” *arXiv Prepr. arXiv2005.09740*, 2020.

[10] W. Jin and C. Florescu, “Improving search and retrieval in digital libraries by leveraging keyphrase extraction systems,” in *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*, 2018, pp. 419–420.

[11] S. Harispe, S. Ranwez, S. Janaqi, and J. Montmain, “Semantic similarity from natural language and ontology analysis,” *Synth. Lect. Hum. Lang. Technol.*, vol. 8, no. 1, pp. 1–254, 2015.

[12] L. Ajallouda, F. Z. Fagroud, A. Zellou, and E. Ben Lahmar, “KP-USE: An Unsupervised Approach for Key-Phrases Extraction from Documents,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 4, pp. 283–289, 2022, doi: 10.14569/IJACSA.2022.0130433.

[13] Y. Zhang, H. Liu, S. Wang, W. H. Ip, W. Fan, and C. Xiao, “Automatic keyphrase extraction using word embeddings,” *Soft Computing*, vol. 24, no. 8. pp. 5593–5608, 2020. doi: 10.1007/s00500-019-03963-y.

[14] K. Bennani-Smires, C. Musat, A. Hossmann, M. Baeriswyl, and M. Jaggi, “Simple unsupervised keyphrase extraction using sentence embeddings,” *arXiv Prepr. arXiv1801.04470,* 2018.

[15] L. Zhang *et al.*, “MDERank: A Masked Document Embedding Rank Approach for Unsupervised Keyphrase Extraction,” *arXiv Prepr. arXiv2110.06651*, 2021.

[16] X. Ye, H. Shen, X. Ma, R. Bunescu, and C. Liu, “From word embeddings to document similarities for improved information retrieval in software

engineering," in *Proceedings of the 38th international conference on software engineering*, 2016, pp. 404–415.

[17] M. Niyogi, K. Ghosh, and A. Bhattacharya, "Learning multilingual embeddings for cross-lingual information retrieval in the presence of topically aligned corpora," *arXiv Prepr. arXiv1804.04475*, 2018.

[18] H. M. Awad and W. M. Saeed, "Semantic Relevance Feedback Based on Local Context," in *2020 International Conference on Computer Science and Software Engineering (CSASE)*, 2020, pp. 296–301.

[19] S. Hofstätter, B. Mitra, H. Zamani, N. Craswell, and A. Hanbury, "Intra-document cascading: Learning to select passages for neural document ranking," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 1349–1358.

[20] S. Siddiqi, "Keyword and Keyphrase Extraction Techniques : A Literature Review Keyword and Keyphrase Extraction Techniques : A Literature Review," no. January, 2015, doi: 10.5120/19161-0607.

[21] X. Zhu, Y. Lou, J. Zhao, W. Gao, and H. Deng, "Generative non-autoregressive unsupervised keyphrase extraction with neural topic modeling," *Eng. Appl. Artif. Intell.*, vol. 120, p. 105934, Apr. 2023, doi: 10.1016/J.ENGAPPAI.2023.105934.

[22] A. R. Mishra, V. K. Panchal, and P. Kumar, "Extractive Text Summarization-An effective approach to extract information from Text," in *2019 International Conference on contemporary Computing and Informatics (IC3I)*, 2019, pp. 252–255.

[23] Y. H. Farhan, S. A. Mohd Noah, M. Mohd, and J. Atwan, “Word-embedding-based query expansion: Incorporating deep averaging networks in Arabic document retrieval,” *J. Inf. Sci.*, p. 01655515211040659, 2021.

[24] D. Jatnika, M. A. Bijaksana, and A. A. Suryani, “Word2vec model analysis for semantic similarities in english words,” *Procedia Comput. Sci.*, vol. 157, pp. 160–167, 2019.

[25] W. Yang, L. Li, Z. Zhang, X. Ren, X. Sun, and B. He, “Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models,” *arXiv Prepr. arXiv2103.15543*, 2021.

[26] S. Dev, T. Li, J. M. Phillips, and V. Srikumar, “On measuring and mitigating biased inferences of word embeddings,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, no. 05, pp. 7659–7666.

[27] G. Glavas, R. Litschko, S. Ruder, and I. Vulic, “How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions,” *arXiv Prepr. arXiv1902.00508*, 2019.

[28] R. Litschko, G. Glavaš, I. Vulic, and L. Dietz, “Evaluating resource-lean cross-lingual embedding models in unsupervised retrieval,” in *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 2019, pp. 1109–1112.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv Prepr. arXiv1301.3781*, 2013.

[30] B. Wang, A. Wang, F. Chen, Y. Wang, and C.-C. J. Kuo, “Evaluating word embedding models: Methods and experimental results,” *APSIPA Trans. signal*

Inf. Process., vol. 8, p. e19, 2019.

[31] X. Rong, “word2vec parameter learning explained,” *arXiv Prepr. arXiv1411.2738*, 2016.

[32] A. Mahmoud and M. Zrigui, “Sentence embedding and convolutional neural network for semantic textual similarity detection in Arabic language,” *Arab. J. Sci. Eng.*, vol. 44, pp. 9263–9274, 2019.

[33] D. Cer *et al.*, “Universal sentence encoder,” *arXiv Prepr. arXiv1803.11175*, 2018.

[34] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *Adv. Neural Inf. Process. Syst.*, vol. 26, 2013.

[35] A. Conneau and D. Kiela, “Senteval: An evaluation toolkit for universal sentence representations,” *arXiv Prepr. arXiv1803.05449*, 2018.

[36] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, “Deep unordered composition rivals syntactic methods for text classification,” in *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, 2015, pp. 1681–1691.

[37] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[38] A. CHAUDHARY, “Universal Sentence Encoder Visually Explained,” *Amitness, Res. Eng.*, 2020.

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

[40] T. Mori and T. Sasaki, “Information gain ratio meets maximal marginal relevance,” *les actes Natl. Inst. Informatics Test Collect. Inf. Retr.*, 2002.

[41] A. Mallia, O. Khattab, T. Suel, and N. Tonellotto, “Learning passage impacts for inverted indexes,” in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 1723–1727.

[42] W. Lu, X. Li, Z. Liu, and Q. Cheng, “How do author-selected keywords function semantically in scientific manuscripts?,” *KO Knowl. Organ.*, vol. 46, no. 6, pp. 403–418, 2019.

[43] V. Bortnikova, I. Nevliudov, I. Botsman, and O. Chala, “Search Query Classification Using Machine Learning for Information Retrieval Systems in Intelligent Manufacturing.,” in *ICTERI*, 2019, pp. 460–465.

[44] L. Zhao, X. Liu, and J. Callan, “WikiQuery-An Interactive Collaboration Interface for Creating, Storing and Sharing Effective CNF Queries.,” in *OSIR@ SIGIR*, 2012, pp. 1–8.

[45] O. Khattab and M. Zaharia, “Colbert: Efficient and effective passage search via contextualized late interaction over bert,” in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 39–48.

[46] S. Plansangket, “New weighting schemes for document ranking and ranked query suggestion.” University of Essex, 2017.

[47] N. T. James and R. Kannan, “A survey on information retrieval models, techniques and applications,” *Int. Journals Adv. Res. Comput. Sci. Softw. Eng. ISSN*, 2017.

[48] K. Park, J. S. Hong, and W. Kim, “A methodology combining cosine similarity with classifier for text classification,” *Appl. Artif. Intell.*, vol. 34, no. 5, pp. 396–411, 2020.

[49] S. MacAvaney, F. M. Nardini, R. Perego, N. Tonellotto, N. Goharian, and O. Frieder, “Efficient document re-ranking for transformers by precomputing term representations,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 49–58.

[50] A. Pawar and V. Mago, “Calculating the similarity between words and sentences using a lexical database and corpus statistics,” *arXiv Prepr. arXiv1802.05667*, 2018.

[51] P. Atanasova, P. Nakov, G. Karadzhov, M. Mohtarami, and G. Da San Martino, “Overview of the CLEF-2019 CheckThat! Lab: Automatic Identification and Verification of Claims. Task 1: Check-Worthiness.,” *CLEF (Working Notes)*, vol. 2380, 2019.

[52] F. Cakir, K. He, X. Xia, B. Kulis, and S. Sclaroff, “Deep metric learning to rank,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1861–1870.

[53] M. Farouk, “Measuring text similarity based on structure and word embedding,” *Cogn. Syst. Res.*, vol. 63, pp. 1–10, 2020.

[54] P. Verma, S. Pal, and H. Om, “A comparative analysis on Hindi and English extractive text summarization,” *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 18, no. 3, pp. 1–39, 2019.

[55] M. A. Rosid, A. S. Fitrani, I. R. I. Astutik, N. I. Mulloh, and H. A. Gozali, “Improving text preprocessing for student complaint document classification using sastrawi,” in *IOP Conference Series: Materials Science and Engineering*, 2020, vol. 874, no. 1, p. 12017.

الخلاصة

نظرًا للتحسينات التقنية والنمو الهائل للبيانات النصية والمصادر الرقمية، هناك تحدٍ لاستخراج كلمات رئيسية عالية الجودة في البحث المعاصر. يعد الاستخراج التلقائي للعبارات الرئيسية أمرًا بالغ الأهمية للعديد من مهام معالجة اللغات الطبيعية واسترجاع المعلومات، بما في ذلك تلخيص المستندات، والتصنيف ، وتوصية المقالة ، وفهرسة النص الكامل .

الهدف من هذا العمل هو الحصول على فهم دلالي للاستعلام والوثائق المفهرسة باستخدام تقنية التضمين لتحسين أداء أنظمة الاسترجاع. الطريقة المقترحة تتكون من عدة مراحل. تتمثل المرحلة الأولى في جمع المستندات في ملف واحد بعد فك ضغط مجلد مجموعة البيانات باستخدام نظام Linux وتحديد المستندات التي لها قرار للاستخدام داخل النظام .تتضمن المرحلة الثانية إجراء المعالجة الاولية للمستندات ثم تم تنفيذ طريقة استخراج العبارة الرئيسية غير الخاضعة للرقابة ، وتخضع العبارات الرئيسية المرشحة المستخرجة لتطبيق مشفر الجملة العالمي (USE) لإنتاج متجهات التضمين مع الاحتفاظ بالمعلومات الأكثر أهمية باستخدام الصلة الهامشية القصوى (MMR). تم تقييم خطوة استخراج العبارات الرئيسية باستخدام مجموعة البيانات (Inspec) التي تحتوي على ملف يدعى (Key) والذي يتضمن (عبارات الهدف أضافها الخبراء لتمثيل كل مستند). تتضمن المرحلة الثالثة الفهرسة والترتيب باستخدام طريقتين، الطريقة الأولى متوسط تضمين العبارة الرئيسية للحصول على درجة المستند، والطريقة الثانية تبني فهرسًا مقلوبًا لحساب وترتيب العبارات الأكثر تشابهًا مع متجه تضمين الاستعلام .كانت المرحلة الرابعة هي تسجيل أكثر الوثائق ذات الصلة بمتجه الاستعلام.

تم تطبيق نموذج الاسترجاع المقترح على مجموعة البيانات .(Fire2011) كانت المرحلة الأخيرة هي نتيجة تقييم (Baseline) ونتيجة نهجي (الفهرسة والتصنيف). باستخدام متوسط الدقة (MAP) كانت نتيجة Baseline (0.61) ، بينما كانت نتيجة نهج متوسط تضمين المستند 0.5589172. وقد لوحظ أن أفضل نتيجة تم تحقيقها مع الفهرس المقلوب التي كانت 0.6277519.

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة بابل

كلية تكنلوجيا المعلومات

قسم البرمجيات

أستخراج تضمين العبارة الرئيسية باستخدام شبكة المتوسط العميق والصلة الهامشية الاكبر لتحسين أسترجاع المعلومات

رسالة مقدمة الى

مجلس كلية تكنلوجيا المعلومات – جامعة بابل كجزء من متطلبات نيل درجة الماجستير في تكنلوجيا المعلومات / قسم البرمجيات

من قبل

علياء عبد الكاظم هادي

بأشراف

أ.د وفاء محمد سعيد

1444هـ 2023م