INTERACTIVE DOCUMENT RETRIEVAL

by

Chetan Borse

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Computer Science

Charlotte

2017

Approved by:

_____
Dr. Wlodek Zadrozny

_____
Dr. Zbigniew W. Ras

_____
Dr. Samira Shaikh

ABSTRACT

CHETAN BORSE. Interactive document retrieval. (Under the direction of DR. WLODEK ZADROZNY)

Traditional Information Retrieval systems present with vast, relevant information as a response to the user search query, which usually consists of different semantic groups. It is a tedious task to look through all the retrieved information, and the user is mostly interested in the post-retrieved documents belonging to one or the other underlying semantic group.

This problem motivated the researchers to provide an Interactive Document Retrieval system to narrow down the search and quickly locate the information. The notion is to identify the semantic groups of documents by clustering the post-retrieved information and to provide the summary for each cluster. In this research, we propose a new approach for document clustering and multi-document summarization. Our new document clustering approach clusters the post-retrieved documents into semantic space of concepts using the document embedding. The document embedding is obtained by the Doce2Vec training on the conceptualized document collection. The proposed approach improves the performance of the document clustering approximately by 6% when compared with the state-of-the-art techniques by considering F-measure. The proposed multi-document summarization technique extracts sentences from the document collection based on the highest importance scores computed using the Lexical Centrality principle. For power iteration, our algorithm uses the sentence embeddings obtained with the PV-DM model. This technique improves the multi-document summarization accuracy nearly by 4% as measured in Rouge-1 metrics. Thus, our new approaches improve the Interactive Document Retrieval framework to the next level.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Dr. Wlodek Zadrozny, for giving me an opportunity to work on this exciting research topic and for providing the invaluable amount of guidance, support, and patience overseeing my research. He is a great mentor and continuously steered me in the right direction.

I would like to thank the committee members, Dr. Zbigniew Ras and Dr. Samira Shaikh, for agreeing to be part of my committee, and providing ideas and feedback. A special thanks to my lab mates Walid Shalaby and Sean Gallagher for their consistent support, guidance, and contribution to this research. I would also like to thank, Dr. Charles Price and University Research Computing, for providing the resources required for the experiments. A big thanks to UNC Charlotte for giving me an opportunity to present this research.

I would like to extend my thanks to my friends Abhishek Bhandwaldar, Tejaswi Konduri, Dheeraj Suvarna, and Kaivalya Vyas for making this journey a pleasant one. Lastly, I sincerely express my profound gratitude to my parents and my sister for their support and encouragement, without whom this accomplishment would not have been possible.

DEDICATION

dedicated to my parents, Lata and Ravindra.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BOW	Bag Of Words.
CERN	European Organization for Nuclear Research.
CRC	Concept Raw Context.
CSIS	Cross-Sentence Informational Subsumption.
HITS	Hyperlink-Induced Topic Search.
HTML	Hypertext Markup Language.
ILP	Integer Linear Programming.
IR	Information Retrieval.
LCS	Longest Common Subsequence.
MMR	Maximal Marginal Relevance.
PCA	Principal Component Analysis.
PV-DM	Distributed Memory Model of Paragraph Vectors.
ROUGE	Recall-Oriented Understudy for Gisting Evaluation.
RSS	Residual Sum of Squares.
TF-IDF	Term Frequency - Inverse Document Frequency.
URL	Uniform Resource Locator.
USPTO	United States Patent and Trademark Office.
WWW	World Wide Web.

CHAPTER 1: INTRODUCTION

1.1 Information Retrieval: State-of-the-art and Challenges

When the Internet was introduced in the early 90s, there were few who could predict the rapid growth of information. To help the users to search useful information, there was a server hosted by the European Organization for Nuclear Research (CERN) which contained a list of available servers on the Internet [1]. But, this centralized index became unfeasible to use for searching the relevant information for a user's need. The method of finding information on the Internet was revolutionized when the first search engines appeared in 1993. By indexing web pages from all over the world, the users could find relevant information by just providing search terms that they were interested in. As a result, some of the most popular search engines like Google, Yahoo, etc. have grown into global multi-billion dollar companies.

With the continuous growth of electronic information (or documents), it has become immensely important to improve existing search techniques for efficient and effective search in a shorter time. Traditional Informational Retrieval systems retrieve and rank documents based on maximizing relevance to the user search query. These systems primarily use term-weighting approaches to retrieve relevant information (or documents). The retrieved documents are then ranked according to its importance determined by Page Rank algorithm (used in Google search engine) or HITS algorithm (used in Teoma, now Ask.com).

The traditional Informational Retrieval systems respond the user query with hundreds or thousands of relevant documents. It is a tiresome task to search through thousands of retrieved results to get the desired document. This happens because of the growing information and the inability of most users to define the appropriate search query.

As per the user study [2], the highest proportion of the search queries contained one or two terms, and only less than 4% of the queries contained more than six terms. Even mobile search queries have 15.5 characters (just over 2 words) as per the Hitwise report.

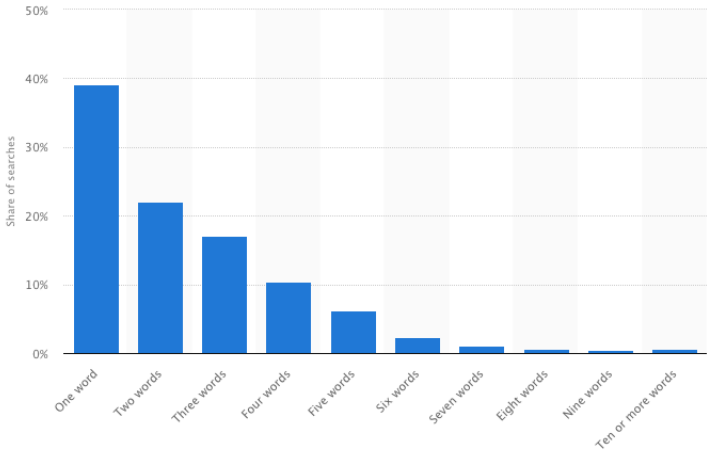

Figure 1.1: Average number of search terms for online search queries in the United States as of July 2017. Credits: statista.com

The voice search has boosted the use of Long Tail keywords, but still average search query length is slightly improved.

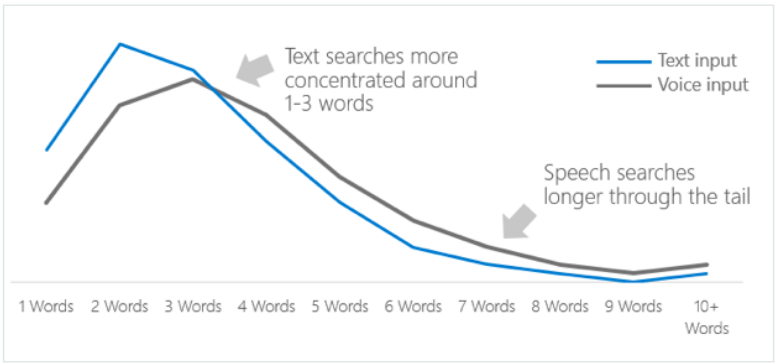

Figure 1.2: Average number of search terms for voice search queries. Credits: moz.com

Further, 77% users looked at one or two pages per search query, which indicates their low tolerance to navigate through a long list of retrieved information (or documents). These facts in themselves stress the need of designing the Information Re-

trieval system in a different way than the systems as practiced to date and bring to the fore importance of the Interactive Information Retrieval (or Interactive Document Retrieval) systems.

1.2 Research Aim

The Interactive Document Retrieval framework aids the user by providing a way to view the search results as the groups of similar documents. The search engines, as Teoma, Vivisimo, and iBoogie, have proven the effectiveness of the search by clustering the post-retrieved documents. The clustering aggregates the similar documents based on the topic, term weight, or other criteria. The goal of the clustering is to provide an overview of the post-retrieved document collection so that a user can quickly locate the desired document. In addition, the multi-document summary can be provided for each cluster to highlight the document similarities and its singular aspect. In this manner, the Interactive Document Retrieval framework reduces the complexity of retrieved information by organizing the retrieved information (or documents) into groups. Just reading the summary of post-retrieved clusters, a user can choose the cluster closer to his or her information needs.

1.2.1 Problem Statement

The existing Interactive Document Retrieval frameworks cluster the post-retrieved documents based on its representation in the vector space using TF-IDF score. TF-IDF technique is based on the bag of words (BOW) model, therefore it loses the ordering and the semantics of the words.

The goal of this research is to propose a new approach that dynamically performs the clustering of the retrieved documents in the semantic space of concepts based on the document embedding (or paragraph vector). Each document is represented by a fixed length, dense vector. These document embeddings are obtained by training the Distributed Memory Model of Paragraph Vectors (PV-DM) [3] on the documents

represented as a bag of concepts. These paragraph vectors work well for tasks that do not have enough labeled data, and preserve the ordering and the semantics of the words. Therefore, the paragraph vectors of the post-retrieved documents boost the performance of the clustering by bringing semantically similar documents together.
The overall objective of this research is to improve the Interactive Document Retrieval framework by clustering the post-retrieved documents based on document embeddings that are obtained from the conceptualized form of the document. Additionally, the new extractive multi-document summarization approach provides the summary for each cluster, which ultimately helps the user to locate the expected document in the short time. In this way, the new document clustering algorithm and multi-document summary approach described in this research, is the great improvement over existing Interactive Document Retrieval frameworks.

1.3 Prior Work

Conventional Information Retrieval system with widely known vector space model was introduced in the early 1970s [4]. With the growth in information (or documents), the document search became tiresome. Consequently, it pointed out the necessity of the revolution in the search engine interfaces. To overcome this inadequacy of the search engine, the researchers in Information Retrieval field proposed the need for an interactive search framework to narrow-down the search and to locate the desired document in minimal time. The notion was to organize the post-retrieved information into groups of similar documents [5] and to summarize each group with its unique aspect [6].
Though the researchers demonstrated the importance of the Interactive Information Retrieval (or Interactive Document Retrieval) system in the late 1990s, the independent research has been started on the document clustering and the document summarization back in the 1960s. Initially, the document clustering was proposed as a method of improving the performance of the Information Retrieval system [7].

In this method, the entire document collection was clustered offline, and the search query was compared with the representation of each cluster. In the late 1990s, the researchers started to apply document clustering technique to the post-retrieved documents [5, 8, 9, 10, 11, 12]. The goal of the post-retrieval clustering is to provide an overview of the retrieved information and to make the document search easier. This idea is also used by search engines, as Teoma, Northern Light and Vivisimo (now acquired by IBM). The user study shows almost half (48.26%) of the post-query records involved displaying result pages that come from clicking on a cluster [13]. It also increased multitasking sessions about 50% longer than the regular search sessions [13]. In addition, the researchers proposed the document clustering based on topic segmentation to identify the cohesive group of segment-based portions of the original documents [14].

The automated multi-document summary for each cluster adds a new value to the Interactive Document Retrieval framework. The work in the automated document summarization dates back in the 1950s, and started at IBM [15]. Some of these single document summarization approaches have been extended to multi-document summarization problem. In the initial attempt of generating multi-document summaries in IR settings, informative summaries are generated with highly structured documents to serve the user's needs in searching [16]. Most summarization techniques adopt the extraction-based approach which selects some original sentences from the group of documents based on heuristics, as the centroid, document title, sentence location, search query [17, 6, 18]. The problem with the extraction-based approach is overlapping information between the selected sentences. So, some researchers proposed compression-based techniques to apply compression on the selected sentences by deleting words or phrases [19, 20, 21, 22, 23]. The most recent trend is the abstraction-based approach which merges the knowledge from different source sentences and mimics the human-written summaries [24, 25].

Our research techniques revolutionize the Interactive Document Retrieval framework by improving the performance of the post-retrieval document clustering approximately by 6%. In addition, we have proposed a new approach based on sentence embeddings, which improves the multi-document summarization and outperforms a LexRank algorithm. In this way, our research helps the user to quickly locate the desired information (or document) just by overlooking the summary of the cluster.

CHAPTER 2: BACKGROUND

2.1 Information Retrieval

Information retrieval (IR) is the activity of obtaining information resources relevant to an information need from a collection of information resources [26]. Searches can be based on full-text or other content-based indexing. Information retrieval is the science of searching for information in a document, searching for documents themselves, and also searching for metadata that describes data, and for databases of texts, images or sounds [26].

The document clustering and document classification are also part of the IR field. Given a document collection, the document clustering is the task to find a good grouping of the documents based on their contents. Given a set of topics, and a document collection, classification is to assign each document to its most suitable topics. The IR systems can be classified as below,

1. IR on the web
2. IR on the document collection
3. IR on a personal computer or laptop

An information retrieval process starts with a user search query. A user search query does not uniquely identify a web page or document. Instead, several web pages or documents may match the user search query with different degree of relevancy. Then, the retrieved results are ranked to finally present it to the user.

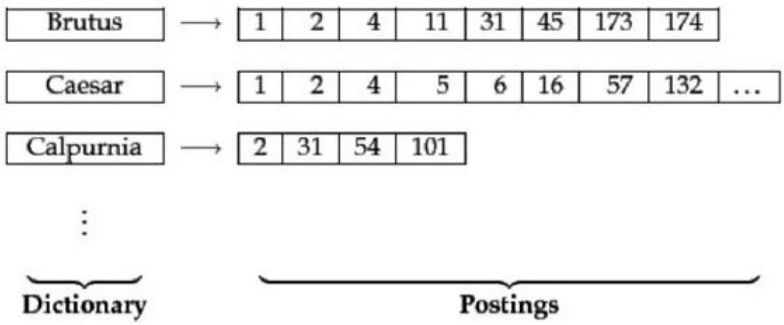

Figure 2.1: The two parts of an inverted index. The dictionary is commonly kept in memory, with pointers to each posting list stored on disk.

The problem with this technique is that its computational cost at query time is more, i.e. linear time in relation to the length of the documents. The solution to this problem is to index the content of the documents like in Figure 2.1. First, an inverted index is built prior to the searches, which speeds-up the searches. An inverted index is composed of a dictionary of all unique terms from the documents. Each entry in the dictionary maps to a list of documents that contain the corresponding term. Such a list of documents is called as a posting list. This inverted index makes it easy and fast to know whether a word appears or not into a document [27].

2.2 Information Retrieval Models

For the effective information retrieval, the documents are first transformed into appropriate representation. Each retrieval technique incorporates a certain model for its document representation. The information retrieval models are categorized based on two aspects, the mathematical basis and the properties of the model. The information retrieval models based on the mathematical aspect are further classified into,

1. Set-theoretic models, which represent documents as sets of words or phrases. The similarities are computed by performing set operations on those sets. e.g. Boolean Retrieval model.

2. Algebraic models, which represent documents and queries as vectors. The similarity between a query vector and the document vectors is computed to find the most relevant documents. e.g. Vector Space model, Latent Semantic Indexing.

3. Probabilistic models, which are based on the probabilistic inference. The similarities are calculated as probabilities that a document is relevant for a given query. e.g. Probabilistic Relevance model (BM25), Language models, Latent Dirichlet Allocation.

4. Feature-based retrieval models, which view documents as the vectors of features and combine these features using learning to rank methods.

2.2.1 Boolean Retrieval Model

The Boolean Retrieval model is the first and most adopted retrieval model. This model is based on the Boolean Logic and Set Theory. In boolean retrieval, each document and query are represented as sets of terms. Information retrieval is based on whether or not the documents contain the provided query terms.
The boolean retrieval model is easy to implement, but it has few disadvantages too. The boolean logic may retrieve too few or too many documents. In the retrieval, all terms are equally weighted. Hence, it is more like Data Retrieval than the Information Retrieval.

2.2.2 Vector Space Model

The Vector Space model is an algebraic model to represent the documents as vectors of terms. It is the base of today's search engines, e.g. Google search engine.
In the vector space model, documents and queries are represented as vectors of weighted terms [27]. The weights can be calculated using different scoring functions like TF-IDF scoring model. The similarity between a document and a query is determined by the cosine of the angle between the vectors, called Cosine Similarity and is calculated as in equation 2.1. The documents that are highly relevant to the

user query, tops the retrieved list of documents with the highest cosine similarity.

$$cos(V_d, V_q) = \frac{V_d \cdot V_q}{||V_d|| \cdot ||V_q||} \tag{2.1}$$

The most popular scoring technique in the vector space model is Term Frequency - Inverse Document Frequency (TF-IDF) [28]. It is a statistical measure of how important a term is to a document in a corpus. The term importance increases proportionally to the number of times term appears in the document, but is offset by the frequency of a term across a collection [28]. TF-IDF is mathematically represented as a product of Term Frequency and Inverse Document Frequency. Term Frequency (TF) is computed as a number of times term occurs in a document and is divided by the document length for normalization as shown in equation 2.2.

$$TF(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d} \tag{2.2}$$

While computing TF, all terms are weighted equally. However, the stop words such as "the", "and", and "an", may appear a lot of times in the document; but in fact, have less importance. Therefore, it is required to reduce the weight of stop words and promote the rare terms. This technique is known as the Inverse Document Frequency (IDF) [28] and is computed as in equation 2.3.

$$IDF(t) = \log \frac{Total\ number\ of\ documents}{Number\ of\ documents\ that\ have\ term\ t} \tag{2.3}$$

The vector space model is a simple model based on linear algebra and overcomes various disadvantages of the boolean retrieval model. However, the vector space model cannot associate the documents with similar context but different term vocabulary. It is a well known bag-of-words model and loses the order in which terms appear. Hence, the user sometimes may not find the desired documents at the top of retrieved

document list. In order to overcome this problem, a couple of information retrieval system started to present the semantic grouping of the post-retrieved documents.

2.3 Page Rank Algorithm

The Page Rank is a link analysis algorithm to measure the relevance of hyperlinked web pages in the World Wide Web (WWW). This algorithm assigns a numerical weight to each document, called as Page Rank of the document [29]. It is the best-known algorithm used by Google to order the search engine results.

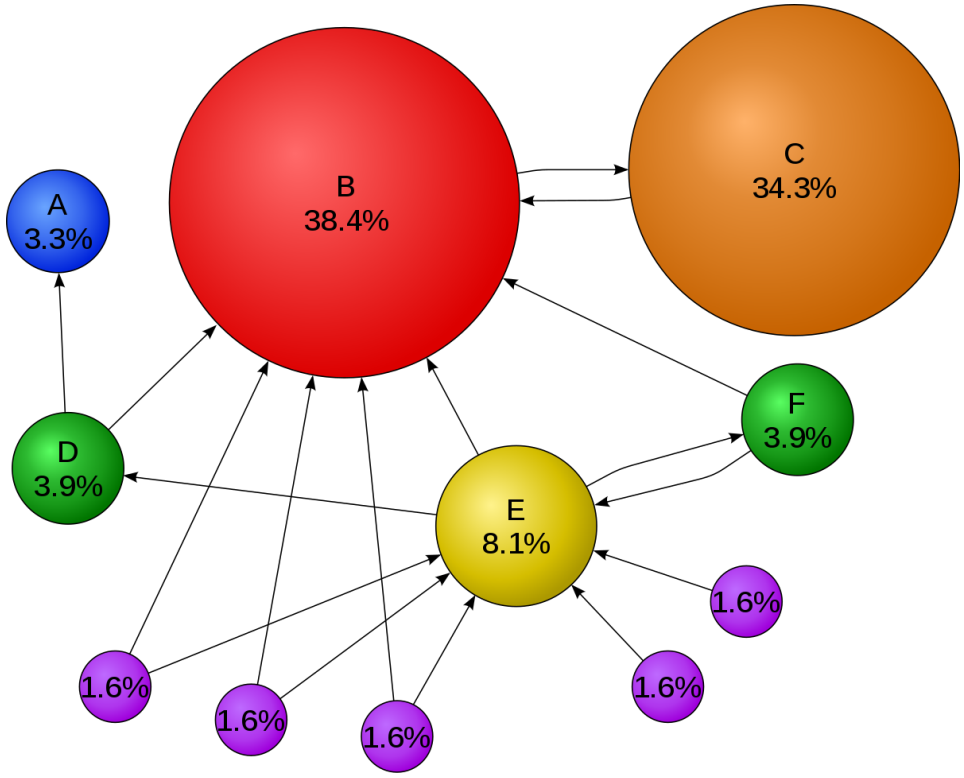

Figure 2.2: Google Page Rank calculation. Credits: en.wikipedia.org

The random surfer visits a web page with a certain probability. The probability of clicking a web page is calculated as a sum of probabilities for a random surfer following links to reach to the page [29]. The surfer does not click on an infinite number of links, but gets bored sometimes and randomly jumps to another page. In order to incorporate this behavior, the page rank algorithm considers the damping factor (d) to compute the probability of reaching to a certain web page. The surfer

always clicks on a random web page with probability (1-d).

Tha page rank equation is given as,

$$PR_i = \frac{1-d}{n} + d \sum_{j \in \{1,\ldots,n\}} \frac{PR_j}{c_j} \tag{2.4}$$

Where PR_i is the page rank of a web page i, PR_j is the page rank contributed by web pages that link to web page i.

2.4 Case Study: Google Search Engine

The amount of information on the web is growing rapidly, as well as the number of people surfing the web is increasing. To address the necessity of information retrieval, Google introduced a search engine which efficiently scales over the large web collection [30].

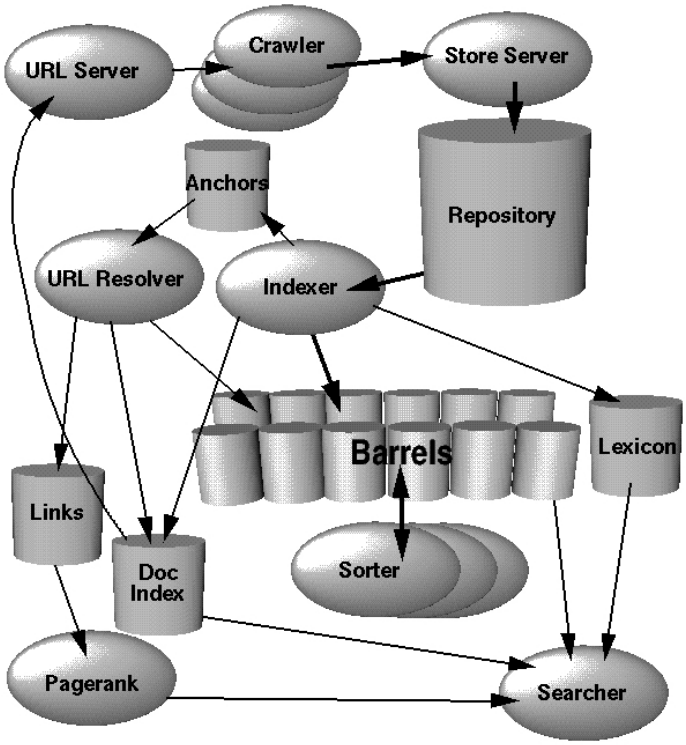

Figure 2.3: High level architecture of Google search engine [30].

In Google search engine, the several distributed crawlers collect data from millions

of web pages. Web pages fetched are sent to the repository that contains the full HTML pages stored in a particular format i.e. (docID, length, URL, and web page) and compressed with zlib. Then, indexer reads repository, uncompresses the documents, and parse them. Each web page is converted into a list of occurrences of terms within a document, called as a Hit List [30]. This hit list has the information such as the term, its position in a document, font size, and capitalization. The indexer stores these hit records into barrels in form of forward index (sorted by termIDs). The indexer also extracts URLs from web pages and store it into anchor file. The URLResolver reads the anchor file and converts relative URLs into absolute URLs to crawl the new web pages. The sorter sorts the barrels by docIDs and generates an inverted index. For every valid termID in an inverted index, lexicon contains a pointer which points to a collection of docIDs that contain the corresponding term. When the user enters a search query, it is tokenized into a list of terms. Then, the tokenized query is matched against an inverted index and the retrieved documents are ranked using the Page Rank algorithm. The Google search engine also considers proximity between terms within a query for improving the search results (phrase queries) [30]. In addition, the Google search engine collects the click data, relevance judgment, and feedback. It then uses the machine learning techniques to personalize the search [30].

2.5 Document Clustering

The Document Clustering is a special field of the Information Retrieval. The idea of document clustering is to assign documents to different topics or topic hierarchies. It is an unsupervised learning, where topic hierarchies are not known in advance. When using clustering in information retrieval, the fundamental assumptions is that the documents in the same cluster behave similarly with respect to relevance to information needs. [27]

The basis for the document clustering is the document representation. The most

commonly used document representation is a Vector Space model, which weighs the terms using TF-IDF scoring function. In Text Mining, the text usually has high dimension. So, the text is first preprocessed using techniques such as the stop word removal, dimensionality reduction (PCA, Latent Semantic Analysis). There are two major categories of the clustering algorithms,

1. Flat Clustering, e.g. K-Means
2. Hierarchical Clustering e.g. Agglomerative, Divisive

2.5.1 Flat Clustering

The Flat Clustering partitions N documents into a set of K clusters that optimizes the chosen partitioning criterion.

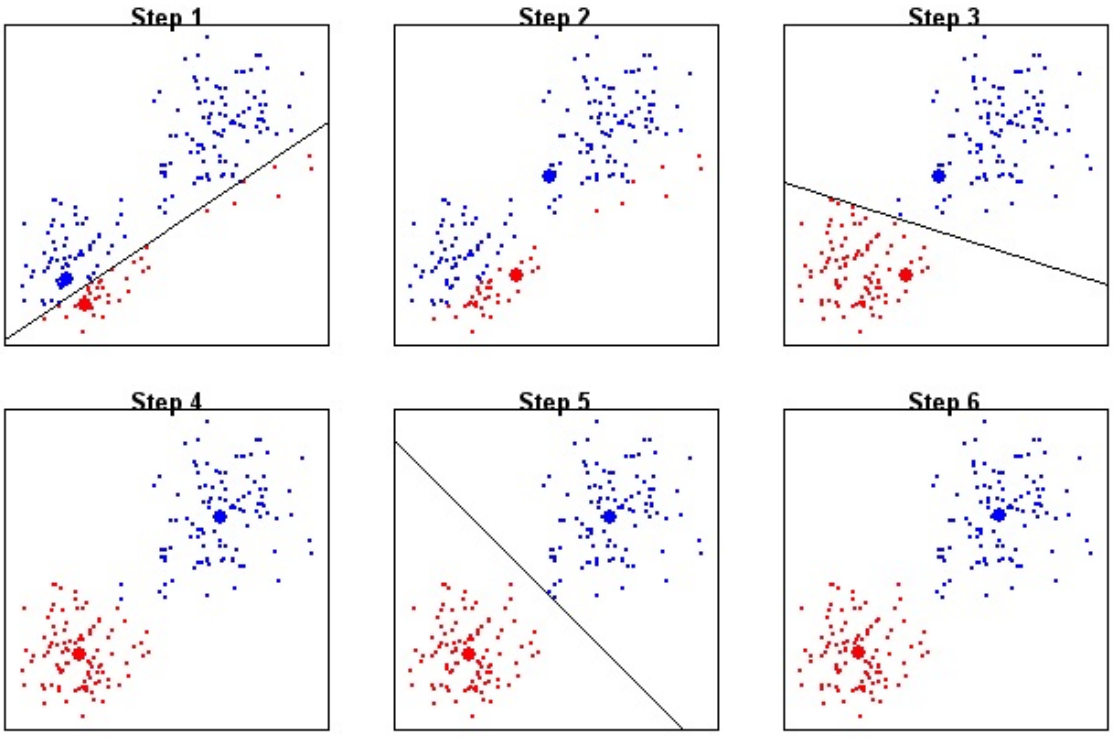

Figure 2.4: K-Means Clustering example. Credits: sherrytowers.com

K-Means is the most popular flat clustering algorithm [31]. Its objective function is to minimize the average squared Euclidean distance of documents from their cluster

centroids [32]. K-Means algorithm first selects K random documents as initial cluster centroids. Then, the algorithm iteratively assigns each document to the cluster with the nearest centroid. It also recomputes the centroid for each cluster based on the documents belonging to the cluster. This iteration continues until it reaches to the stopping criterion, such as the Residual Sum of Squares (RSS) falling below the threshold. The equation for residual sum of squares is shown as below,

$$RSS = \sum (y - \hat{y})^2 \tag{2.5}$$

2.5.2 Hierarchical Clustering

The Hierarchical Clustering is a cluster analysis technique which builds a hierarchy of clusters. The hierarchical clustering generally falls into two categories [33],

1. Agglomerative
2. Divisive

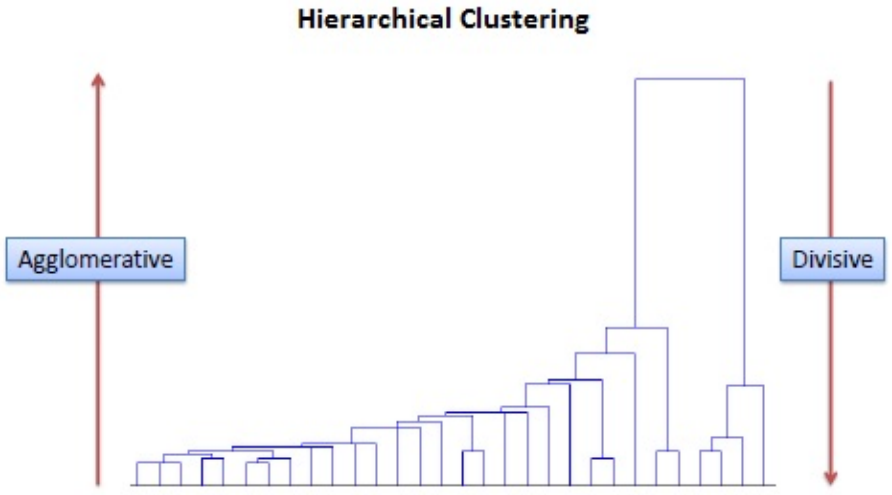

Figure 2.5: An example of Hierarchical Clustering.

The Agglomerative Clustering is a bottom-up approach as shown in Figure 2.5.

Each data point starts as an own cluster and pairs of clusters are merged as algorithm moves up the hierarchy. The time complexity of agglomerative clustering is $\mathcal{O}(n^2 \log(n))$, which makes the algorithm very slow for large data corpus.

The Divisive Clustering is a top-down approach as shown in Figure 2.5. This algorithm starts with all data points in one cluster and then splits the cluster recursively as algorithm moves down the hierarchy. The divisive clustering has even worse time complexity i.e. $\mathcal{O}(2^n)$.

2.6 Distributed Representation of Document

Many machine learning algorithms such as document clustering require the input to be represented as a fixed-length feature vector. The most common feature vector representation is the bag-of-words or bag-of-n-grams [34]. The popularity of bag-of-words representation is due to its simplicity and often surprising accuracy. Despite its popularity, bag-of-words has two major weaknesses. It loses the ordering of words in a document and also discards the semantics of the words. However, bag-of-n-grams gives better results than bag-of-words representation, it is suffered from high dimensionality of the embedding. The better technique is the Distributed Memory Model of Paragraph Vectors (PV-DM).

PV-DM is an unsupervised algorithm to learn the fixed-length, dense vector representation for a document, paragraph or sentence [3]. This dense vector representation overcomes the weakness of bag-of-words and bag-of-n-grams model. The paragraph vector is averaged or concatenated with other word vectors to predict the next word in a context. The paragraph vector can be thought as a memory that remembers what is missing in the current context. These paragraph vectors perform well and achieve the great improvement over other feature vector representations.

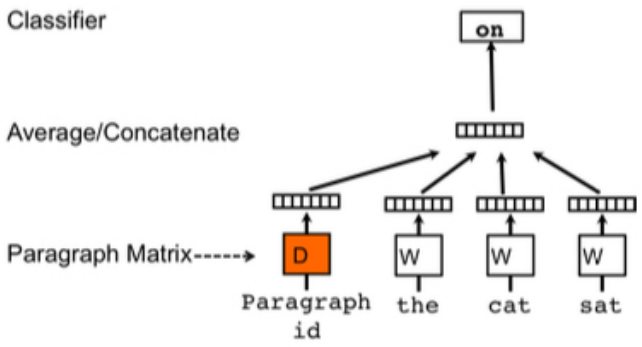

Figure 2.6: A framework for learning paragraph vector [3].

In PV-DM, the paragraph vectors and word vectors are trained using stochastic gradient descent and the gradient is obtained via backpropagation. At every step of stochastic gradient descent, the algorithm computes the error gradient from the network in Figure 2.6 and use the gradient to update the parameters in the model [3]. At prediction time, an inference step is performed to generate the paragraph vector for a new paragraph or document. This is also computed by gradient descent with the fixed parameters for the rest of the model, the word vectors W and the softmax weights [3].

Once trained, these dense vectors can be used as features representation for the paragraphs or documents, and can be fed to conventional machine learning techniques such as Logistic Regression, Support Vector Machines or K-Means.

CHAPTER 3: INTERACTIVE DOCUMENT RETRIEVAL FRAMEWORK

As per the study [35], the mean number of terms per search query is 2.6. So, the large proportion of the search queries contains a maximum of two terms. This is not only due to the broadness of search needs but also because of the inability of the users not to define the appropriate search query. So usually, the impact of such shorter queries is a large collection of the retrieved documents, most of them without any semantic relation to the query. The users also do not have the interest to look through the entire collection of the retrieved documents. Additionally, such a retrieved document collection may contain many underlying topics and the user may be interested only in a certain topic. So, we need an interactive interface to narrow-down the search as per the user's selection of the post-retrieved document clusters. Several researchers have proposed and implemented the Interactive Document Retrieval framework. The main goal of such an interactive document retrieval framework is to find the groups of similar documents and summarize every group. For effective search, this technique is iteratively applied to the post-retrieved results and then user picks the relevant cluster/s at each time. Such interactivity in search systems reduces the search efforts and improves the quality of search results. One of the best examples of such interactive retrieval systems is its application to health domain [36]. People and patients increasingly use the internet to search for health information. However, they face problems searching the exact medical term in the information system. Searching the right medical term requires iterative search and query reformulation [37], which may be difficult for laypersons who do not have much knowledge about health domain. So, the interactive retrieval system can solve this problem by clustering post-retrieved results. Then, people can pick cluster/s of their choice just

by reading the short summary of each cluster. The evaluation of interactive search system proves its efficacy. We further leverage this interactive search framework by proposing the new techniques for document clustering and multi-document summarization. Such improved framework as shown in Figure 3.1 will further reduce the search efforts and help the users to locate the expected search results in shorter time.

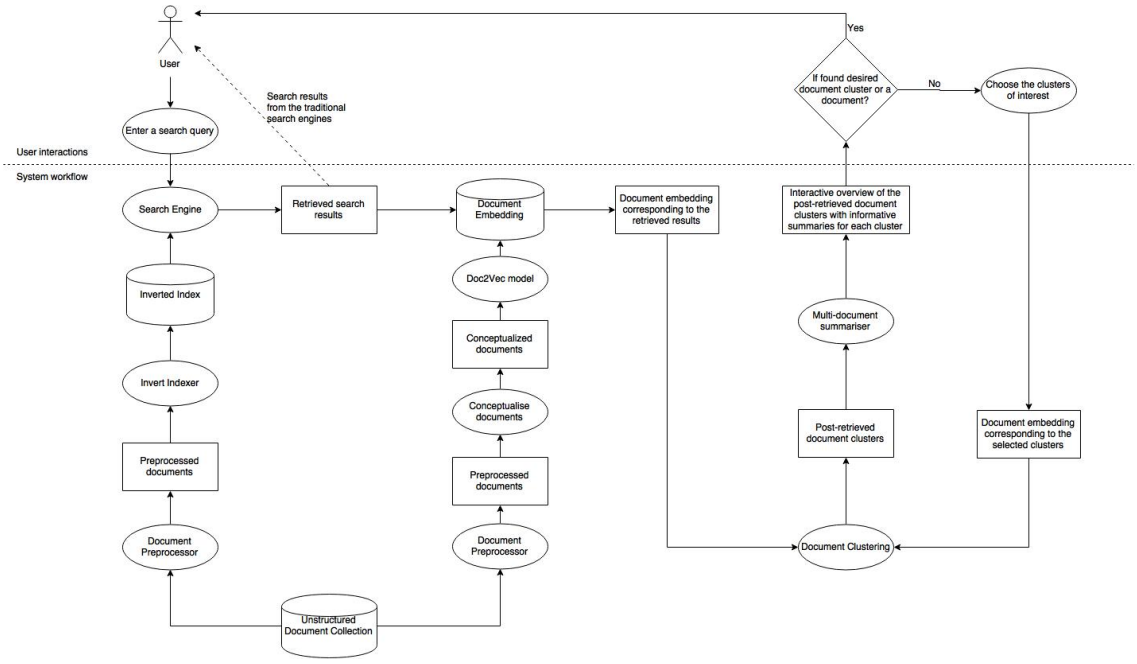

Figure 3.1: An Interactive Document Retrieval framework.

The proposed Interactive Document Retrieval framework has two major components, the Document Clustering, and the Multi-document Summarization. The document clustering part of the system finds the semantic groups of documents within the post-retrieved results. The basis for such document clustering technique is the fixed-length document embedding generated for each document offline. We boost the performance of our document clustering technique by conceptualizing the document collection even before generating the document embeddings [38]. Then, the Distributed Memory Model of Paragraph Vector (PV-DM) is trained on the conceptualized document collection to generate the fixed-length feature vector for each variable-length document. After generating the document embeddings offline, we use

the Bisecting K-Means algorithm to cluster the post-retrieved documents in real-time [39]. Our proposed document clustering technique improves clustering performance over the state-of-the-art approaches. Our algorithm achieves the significant gain of 6% in F-measure.

In order to assist the users, the multi-document summary is generated for every cluster. Our multi-document summarization technique is based on extraction-based approach and summarizes the contents of every cluster with the fixed number of sentences. The root of the multi-document summarization algorithm is the sentence embeddings. The PV-DM model is trained on the sentence collection to generate the sentence embeddings. At runtime, the cosine similarity matrix is created by computing the similarity between a pair of sentence embeddings. We use the principle of the Lexical Centrality and run the power iteration over the cosine similarity matrix [17]. This approach provides the centrality measure for every sentence. Thus, the algorithm picks the top sentences, compress them using compression techniques to remove unimportant words and phrases, and generates the final synthesized summary. When the user enters a search query to retrieve the documents of his or her interest, the traditional search engines provides a long list of relevant documents. On the other hand, the interactive document retrieval framework presents the interactive view of the post-retrieved document clusters with the individual cluster summary. Thus, the user makes a choice of cluster/s. If he or she finds the desired document in the selected cluster/s, then the algorithm stops. Otherwise, the document clustering and the multi-document summarization iteratively happen with the user-selected cluster/s.

CHAPTER 4: DOCUMENT CLUSTERING

The document clustering algorithm produces the groups of cohesive documents, where each cluster has its unique aspect or topic. One of the well-known examples of such document clustering systems is the Noggle Knowledge Assistant as shown in Figure 4.1. In general, the clustering is performed on documents that are represented using the vector space model. The traditional document clustering methods use K-Means or Hierarchical clustering techniques to cluster the document feature vectors. As these document feature vectors are based upon TF-IDF model which a bag-of-words model, the document representation loses the word ordering and even semantics. Ultimately, the document clustering does not produce the desired results.

Figure 4.1: An example of Document Clustering system (Noggle Knowledge Assistant). Credits: noggle.online

4.1 Algorithm

We propose a new document clustering algorithm based on document embeddings. Each document is represented by the corresponding document embedding. These document embeddings are obtained by training the Doc2Vec model on the conceptualized document collection. Then, the document clustering is performed on document embeddings using the Bisecting K-means algorithm [39]. This new approach generates the document embeddings offline and performs the document clustering in a real time.

Algorithm 1: The Proposed Document Clustering Approach

1. Preprocess the document collection.
2. Conceptualize the document collection.
3. Train Doc2Vec model on the document collection and generate corresponding document embeddings offline.
4. Once the user enters a search query and system retrieves the relevant results, then get back the corresponding embeddings through database lookup.
5. Perform Bisecting K-Means clustering on retrieved embeddings and find the document clusters n, as specified by the user.

4.2 Document Embedding

The basis for document clustering is the document embedding which is a fixed-length representation for the variable-length document. The process of generating the document embedding has three major steps, preprocessing, conceptualization and Doc2Vec training.

4.2.1 Preprocessing

The terms in documents often have many structural variants. For this reason, the document preprocessing is required to remove such variants and increase the effectiveness of information retrieval [40].

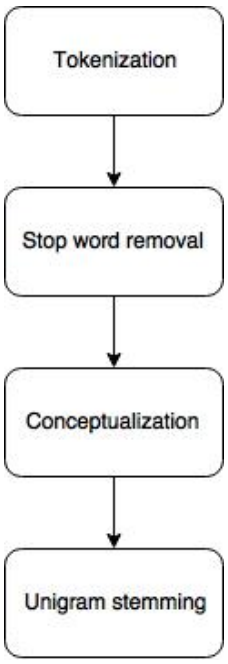

Figure 4.2: Steps for preprocessing the documents.

First, the documents are tokenized into individual words. Then, stop words are removed, as they are less important in information retrieval and make the text heavier. Removing stop words reduces the dimensionality of term space. In the next step, the documents are conceptualized with possible concept length up to eight terms. Our document clustering technique uses the concept embedding model for conceptualizing the document collection [38].

Additionally, all unigrams are lemmatized based on its part of speech tag. The lemmatization is the process of removing inflectional endings by using vocabulary and morphological analysis of words. It usually provides the dictionary (base) form of a word, known as the lemma. We prefer the lemmatization over stemming, because the stemming process is a crude heuristic and often chop off derivational affixes too. The exceptional benefit of lemmatization is that it selects the appropriate lemma when the word context is provided. Also, the lemmatization process preserves the meaning of a word. In our approach, we first identify the part-of-speech tag of a word and provides it as a context to the lemmatizer. In the research, we use nltk's WordNetLemmatizer

for lemmatization and PerceptronTagger for part-of-speech tagging.

4.2.2 Conceptualization

The idea behind the document conceptualization is to transform the textual structures in a document into a semantic space of concepts which captures the main topic of these structures [38]. Our algorithm conceptualizes every document with a dense bag-of-concepts.

In preprocessing the document collection, the algorithm uses the pre-trained Concept Raw Context (CRC) model. The CRC model jointly learns the embeddings for words and concepts. By considering both concepts and individual words in the optimization function, the algorithm generates more robust and high-quality embeddings. Subsequently, these concept embeddings obtained by from the document corpus are fed to Doc2Vec model. It ultimately boosts the performance of Doc2Vec model and document clustering in the later steps.

4.2.3 Doc2Vec Training

The document clustering requires the documents to be represented as fixed-length vectors. The most common document representation is a bag-of-words model or bag-of-n-grams model [34], because it is simple and often provides surprising accuracy. Unfortunately, the bag-of-words representation loses the word ordering and semantics of the text. On the other hand, the bag-of-n-grams model is suffered from the high dimensionality of the embedding. The better approach is to use the Distributed Memory Model of Paragraph Vectors (PV-DM). The PV-DM model learns the continuous distributed vector representations for pieces of texts [3]. These paragraph vectors can be used as the feature vector representation for the document collection and can be fed to the document clustering. The notion of using the paragraph vectors over bag-of-words model is to preserve the meaning and the order of words. It ultimately helps to bring the documents with same semantic contents closer. As a result, the

document clustering yields better grouping of the document collection.

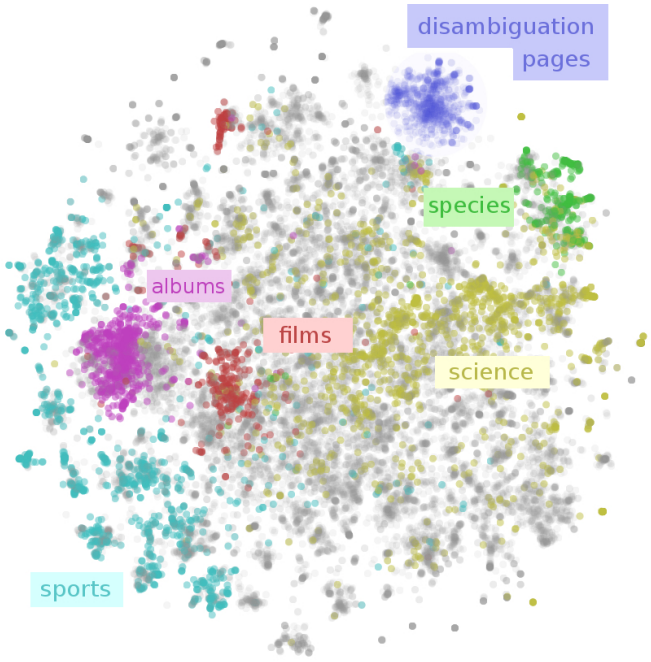

Figure 4.3: Visualization of the Paragraph Vectors. Credits: Colah's blog

The major step in the process of generating document embedding is the Doc2Vec training. First, the Doc2Vec model is created with a predefined set of hyper-parameters. While setting up hyper-parameters for the Doc2Vec model, the algorithm discards the words with term frequency less than the minimum term frequency threshold. In this way, the overall size of the vocabulary can be reduced. The model also considers the negative sampling to add the noise for generalization. The next step is to build the vocabulary of unique n-gram concepts from the document collection. Then, this vocabulary is intersected with the pre-trained concept embedding [38]. The intersection step merges the input-hidden weight matrix from the pre-trained concept embedding and initializes the weights of the concepts in vocabulary.

Algorithm 2: Doc2Vec Training

1. Create a Doc2Vec model with embedding size 500, minimal term frequency 5, context window size 8, downsampling 1e-5, and negative sampling.
2. Build a vocabulary of concepts from the given document collection.
3. Intersect the vocabulary with pre-trained concept embedding.
4. Train Doc2Vec model with Stochastic Gradient Descent algorithm by shuffling the document collection after every pass.

The Doc2Vec model is trained using the Distributed Memory Model of Paragraph Vectors (PV-DM) technique [3]. Our approach also shuffles the document collection after every pass for generalizing the trained model. Once the Doc2Vec model is trained, the document embedding is obtained by transforming the memory-mapped document vectors (trained). The document embedding for an unseen document is obtained by performing an inference step with gradient descent on a new, unseen document.

The goal is to generate the document embeddings for a given document collection. We generate these document embeddings offline and save them into the structured database. Later, as per the user search query, the embeddings corresponding to the post-retrieved documents are obtained back by performing the database lookup. Next, these retrieved embeddings are fed to the clustering algorithm to find the inherent groups in post-retrieved documents.

4.3 Document Clustering Approach

The document clustering algorithm splits the document collection among groups of similar documents.

According to the nature of clusters it produces, the document clustering has two different categories, Partitional (Flat) and Hierarchical clustering [41]. The partitional clustering technique such as K-Means clustering simply divides the document collection into a pre-defined number of clusters. Every cluster obtained by K-Means

clustering is represented with a centroid. The cluster centroid is a representative of the set of documents within a cluster. On the other hand, the hierarchical clustering produces a nested set of document partitions that can be visualized as a tree or dendrogram. The leaves of the tree represent the documents. This tree can be generated either bottom-up or top-down. Bottom-up (Agglomerative) clustering starts with the individual documents and groups the most similar document at every step. Top-down (Divisive) clustering starts with an entire document collection and divides them at each step to maximize the similarity within the cluster. The advantage of K-Means clustering is its linear time complexity. However, the benefit of Hierarchical clustering is the dendrogram that corresponds to a meaningful taxonomy. Our research uses a combination of partitional and divisive clustering, called as the Bisecting K-Means clustering [39].

4.3.1 Bisecting K-Means Clustering

The Bisecting K-Means clustering combines the strengths of both types of clustering. This type of clustering starts with a single cluster containing all the documents. In each iteration, a cluster to split is selected based on a certain criterion. The criterion to split the cluster can be the size of the cluster or maximize the overall similarity. Then, the K-Means clustering (K=2) is applied to selected cluster for splitting it into two separate groups. This bisecting step is repeated until the desired number of clusters with the highest overall similarity are obtained.

There are different clustering criterion functions such as I_2, ϵ_1, and H_2. I_2 criterion function maximizes the similarity between each document and the centroid of the cluster to which it is assigned to. ϵ_1 function minimizes the cosine between the centroid of each cluster and the centroid of the entire document collection. H_2 is a hybrid function of I_2 and ϵ_1.

$$maximize\ I_2 = \sum_{r=1}^{k} \sum_{d_i \in S_r} cos(d_i, C_r) \tag{4.1}$$

$$minimize\ \epsilon_1 = \sum_{r=1}^{k} n_r \frac{D_r^t D}{||D_r||} \tag{4.2}$$

$$maximize\ H_2 = \frac{I_2}{\epsilon_1} \tag{4.3}$$

The Bisecting K-Means clustering with the criterion function I_2 has better accuracy than the K-Means and the Hierarchical clustering [6]. Our research uses the Cluto library for the Bisecting K-Means clustering with the I_2 criterion function [42, 43]. The Cluto library is written in C for clustering the high-dimensional data in short time. Therefore, this library speeds up our algorithm and the document clustering happens quickly.

CHAPTER 5: MULTI-DOCUMENT SUMMARIZATION

The Multi-document Summarization is an automatic process of extracting information from multiple texts (or documents) for the same topic. The users read such multi-document summary and quickly get familiar with the topic. The goal of multi-document summarization is to simplify information search and reduce the time by pointing to the most relevant documents.

The multi-document summarization is more complex than a single document summarization. The main problem with multi-document summarization is the potential redundancy. Ideally, the multi-document summary should contain both 'central' and 'diverse' information. There are a couple of state-of-the-art techniques that address this particular requirement. e.g. LexRank [17], Maximal Marginal Relevance (MMR) [44], GRASSHOPPER [45].

Below are the different ways to generate the multi-document summary,

1. Extraction-based summarization: This approach extracts whole sentences from the text without modifying them and creates a short summary. e.g. LexRank.

2. Abstraction-based summarization: This approach involves the paraphrasing sections of the text. It condenses a text more strongly than extraction, but the application that can do it is hard to develop.

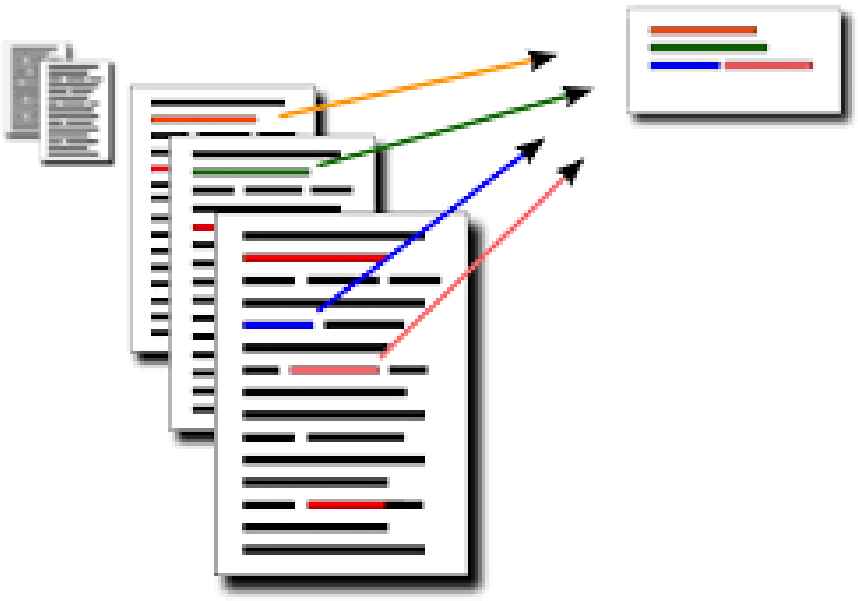

Figure 5.1: An idea of Multi-document Summarization based on the sentence extraction.

The majority of multi-document summarization systems are extractive, e.g. Sentence centrality based techniques, LexRank. The LexRank estimates the sentence importance using random walks and eigenvector centrality [17]. It constructs a graph by creating a vertex for each sentence in the document. The edges between sentences are simply based on semantic similarity. The LexRank creates a cosine similarity matrix of TF-IDF vectors and applies the power iteration to rank the sentences. Finally, it forms the summary by combining the top ranking sentences. The LexRank is a part of the large summarization system, named MEAD [46].

5.1 Algorithm

Our multi-document summarization approach is extraction-based and uses the LexRank as a base model.

Our approach first trains the Sentence2Vec model on the corpus of sentences. The Sentence2Vec model generates the embeddings for sentences in the document clusters. When the document clustering algorithm clusters the post-retrieved documents,

our multi-document summarization algorithm tokenizes every document cluster into sentences and generates the embeddings for them using the Sentence2Vec model. It then creates a cosine similarity matrix and runs a power iteration over it to compute the importance scores for the sentences. Finally, it picks the top sentences based on the ranking score and adds them to the summary. This new approach trains the Sentence2Vec model offline and performs the power iteration on cosine similarity matrix at run-time.

This new multi-document summarization uses the sentence embedding, which is obtained using the Doc2Vec training and is a good semantic representation when compared with TF-IDF representation. Ultimately, the summary obtained through our algorithm is more meaningful and mimics the human-generated summary.

Algorithm 3: The Proposed Multi-document Summarization Approach

1. Preprocess the document collection.
2. Train the Sentence2Vec model on sentences from the corpus.
3. Use trained Sentence2Vec model to generate sentence embeddings for the sentences in the post-retrieved documents.
4. Create a cosine similarity matrix for the similarity between a pair of sentences.
5. Use the Lexical Centrality principle and run a power iteration over cosine similarity matrix to calculate the importance scores for the sentences.
6. Pick the top sentences based on the importance scores and include them in the summary. (Note: Exclude the duplicate or most similar sentences based on the similarity socres.) Post-process sentences in summary using Clarke & Lapata (2008)'s ILP model and generate the final compact summary [47].

5.2 Sentence2Vec Model

Traditional extraction-based techniques for multi-document summarization use the TF-IDF vectors to represent the documents. The TF-IDF model loses the word ordering and the semantics. It is a sparse vector representation for sentences. Therefore,

our approach uses the sentence embeddings obtained by training the Doc2Vec model on the sentence corpus. These sentence embeddings are smaller in dimension and preserve the word order. Thus, it is a good representation for computing the sentence similarity. Our approach first trains the Sentence2Vec model. It is a two-step process that involves the sentence preprocessing and Sentence2Vec training.

1. Sentence Preprocessing: All documents in the cluster are tokenized into sentences. Every sentence is preprocessed by stop-word removal and stemming the unigrams. We use the similar preprocessing techniques as discussed in the document clustering approach.

2. Sentence2Vec Training: The Sentence2Vec model is trained on the sentence collection. We use the Distributed Memory Model of Paragraph Vectors (PV-DM) algorithm to train the Sentence2Vec model [3].

The idea behind the Sentence2Vec model is to generate the sentence embeddings that preserves the semantics of sentences. Our approach creates the Sentence2Vec model with the predefined set of hyperparameters, such as embedding size, minimum term frequency, and negative sampling, etc. It builds the vocabulary of unique words and intersects with the pre-trained word2vec embedding. Finally, it trains the Sentence2Vec model by shuffling sentences in the corpus.

The proposed multi-document summarization algorithm uses the trained Sentence2Vec model to generate the embeddings for sentences at run-time. These embeddings are further used to produce the multi-document summary for a cluster of documents.

5.3 Sentence Extraction

The next step in our approach is the sentence extraction based on its importance in the cluster of documents. The importance of sentence is determined by how much the common information the sentence has.

Our approach uses the graph-based technique to extract sentences from the corpus and

is inspired by the LexRank algorithm. Initially, the graph of the sentence similarity is created. It treats the sentences as vertices and the similarity relation between sentences as edges. The motive of building a graph of sentences is to find the most connected sentence in the set of documents. We achieve this goal using the Lexical Centrality principle.

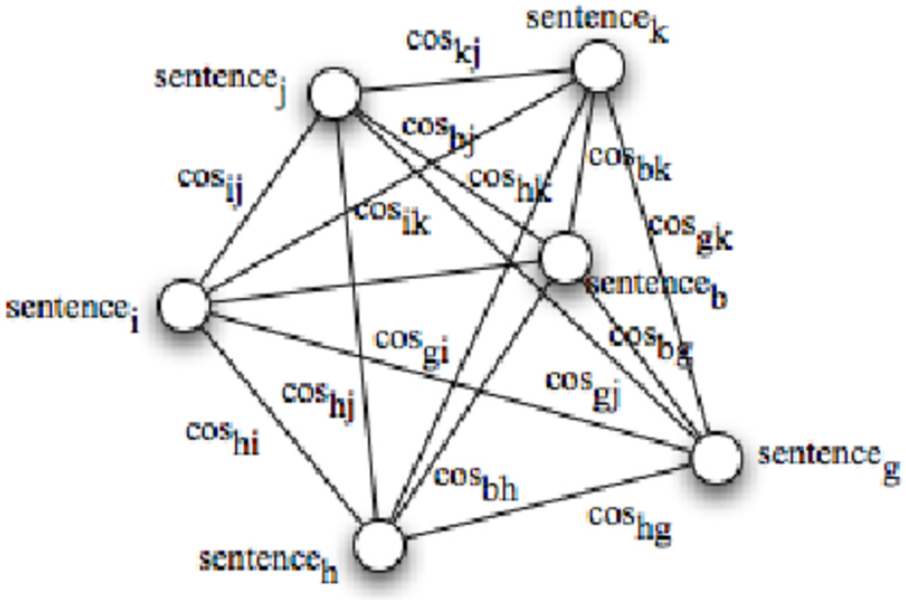

Figure 5.2: A graph of sentences with sentences as vertices and cosine similarity between a pair of sentences as an edge [48].

First, the cosine similarity matrix is created by computing the similarity between a pair of sentences. Then, the power iteration iteratively computes the sentence importance score by running the power iteration over a cosine similarity matrix [17].

$$CentralityMeasure(u) = \frac{d}{N} + (1 \ - \ d) \sum_{v \in \{adj[u]\}} \frac{CentralityMeasure(v)}{degree(v)} \qquad (5.1)$$

Once the convergence happens, then the power iteration terminates with the importance score for every sentence in the corpus. Finally, the algorithm sorts the sentences in the decreasing order of its importance scores. The top sentences with the highest

importance scores are included in the final summary. Additionally, the sentences in summary are sorted as per their order in the sentence corpus, so that the information remains in an original flow and does not deviate from the common topic of a cluster of documents.

Once the summary that has a common information for the cluster of documents is extracted, then it is further post-processed using techniques such as the sentence compression, CSIS, etc. to generate the human-like summary.

5.4 Summary Post-processing

The post-processing is performed on the extracted sentences in order to get the compact, diverse, and well-formed summary.

1. Remove Duplicate Sentences: The duplicate or most similar sentences are removed from the summary for synthesizing diverse multi-document summary that covers multiple documents in the cluster. Our approach computes the cosine similarity between a new sentence to be added to summary and the sentences that already exist in the summary. It adds a new sentence in the summary only if the cosine similarity is less than the threshold 0.98. Thus, this technique formulates the multi-document summary that covers diversive knowledge from the document cluster.

2. Sentence Compression: This algorithm optionally post-process the multi-document summary by sentence-level compressions via deletion. It uses the implementation based on the ILP model as described in [47].

CHAPTER 6: EXPERIMENT

The experiments conducted in this research are presented in this chapter. The experiments are performed to better understand the behavior of the proposed document clustering and multi-document summarization techniques. The motive of the experimental setup is to evaluate the proposed techniques. It analyses how the post-retrieved document clustering and multi-document summarization algorithms increase the user's ability to narrow-down search and his or her understanding about the different aspects of post-retrieved information.

6.1 Dataset

The experiments are conducted primarily on the 20 NewsGroups dataset and Opinosis dataset. The 20 NewsGroups dataset is a state-of-the-art dataset for the document clustering [49]. The Opinosis dataset is widely used for evaluating the summarization algorithm [50]. Few experiments are also performed on BBC news articles hosted in the University College Dublin [51] and Patent dataset hosted by United States Patent and Trademark Office (USPTO).

6.1.1 20 NewsGroups

The 20 NewsGroups dataset is a collection of approximately 20,000 newsgroup documents and constitutes twenty different categories as described in Figure 6.1. It also has six different high-level categories, namely religion, politics, science, computer science, sport, and sales ads. The document collection in the 20 NewsGroups dataset is highly unstructured and differs significantly in lengths. This dataset contains some HTML meta tags and requires the data cleaning. The dataset can be downloaded at http://qwone.com/ jason/20Newsgroups/.

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Figure 6.1: The 20 News Groups dataset.

6.1.2 Opinosis - Topic related review sentences

The Opinosis dataset contains sentences extracted from user reviews (from various sources like Tripadvisor, Edmunds.com, and Amazon.com).
It consists of 51 different topics with each topic having approximately 100 sentences. Example topics are "battery-life of iPod nano" and "sound quality of iPod nano", etc. This dataset also comes with gold standard summaries ($\sim$ five per topic) which can be used to evaluate the summarization algorithm.

6.1.3 BBC News Articles

The BBC News Articles are the original articles owned by the BBC. It consists of 2225 documents from the BBC news website corresponding to stories in five topical areas from 2004-2005.
It covers topical areas like business, entertainment, politics, sport, tech, etc.

6.1.4 Patent Collection

The Patent data set is a huge collection of nearly 7.5 million patent documents from the United States Patent and Trademark Office (USPTO). This document collection is in the unstructured format, but every document has a set of known fields, such as title, abstract, description, claims, etc.

6.2 Experimental Setup

The experiments are performed on dataset mentioned above in different phases. The independent and combined experiments for the document clustering and multi-document summarization are performed.

6.2.1 Document Clustering

For the document clustering experiments, we first preprocess the 20 NewsGroups data collection and then generate the embeddings offline. The 20 NewsGroups dataset is clustered among 20 document clusters for evaluating our document clustering technique. The document clustering algorithm is run on the document embeddings of different dimensions, such as 150, 200, 500. We also perform experiments with and without the document embedding. Finally, the top clustering accuracy is compared with the other state-of-the-art techniques.

6.2.2 Multi-document Summarization

For the multi-document summarization, we preprocess the Opinosis dataset and train the Sentence2Vec model using the collection of sentences from the dataset.
When summarizing the articles, we first tokenize the collection into sentences and generate corresponding sentence embeddings. Then, the cosine similarity matrix is built using the similarity between every pair of sentences. Next, the power iteration is run on the cosine similarity matrix and the sentence importance score is computed for each sentence. According to the summary length required, the algorithm picks top sentences with highest scores.
While adding sentences into the final summary, it also removes duplicate or highly similar sentences based on the similarity threshold. It also post-processes the summary to compress the sentences and to generate the compact summary.
We perform experiments on the Opinosis dataset and evaluate our multi-document summarization approach using different metrics such as cosine similarity, rouge, etc.

We also compare our multi-document summarization algorithm with other state-of-the-art extraction-based techniques.

6.2.3 Interactive Document Retrieval

For experimenting the document clustering and the multi-document summarization together, we first create an inverted index over BBC News Articles. We also generate the document embeddings for BBC News Articles offline and store it in the database. Additionally, Sentence2Vec model is trained offline on the sentence corpus to generate the embeddings for sentences.

At run-time, when the user enters a search query, the relevant articles are retrieved using traditional TF-IDF approach. These retrieved results are fed to the document clustering algorithm, which identifies the groups among the retrieved results. Every cluster is also summarized using the extraction-based algorithm. For generating the multi-document summary, we pick 25 documents from each document cluster and generate the multi-document summary. The document selection for the multi-document summary is accomplished based on the fact how close the documents are to the cluster centroid. The reason for generating the multi-document summary using the selective documents is to reduce the computation time and to quickly present the cluster summary.

Thus, the user is provided with these document clusters along with their summaries. Then, the user makes a choice of the desired cluster. The documents within chosen clusters again re-clustered to get the new groups and summaries. In this way, the search results are optimized in every iteration and the user can find the documents of their exact requirements.

For verifying the effectiveness of the interactive search system, we ran a couple of search queries and manually assessed whether the desired documents are retrieved or not.

6.3 Evaluation Metrics

Different evaluation metrics are used to assess the proposed algorithm for document clustering and multi-document summarization. The proposed techniques are independently evaluated and compared with the respective state-of-the-art techniques.

The quality of the document clusters can be inspected by using either internal quality measure or external quality measure. The internal quality measure allows comparing different document clusters without reference to external knowledge. The external quality measure evaluates the document clustering approach by comparing the produced clusters to known classes. We use three different external measures, Purity, Entropy [52], and F-measure [53].

Precision, Recall, Cosine Similarity, and ROUGE are some of the metrics which are used to evaluate the multi-document summaries.

6.3.1 Precision

Precision is the fraction of the documents retrieved that are relevant to the user's information requirement.

$$Precision = \frac{|Relevant\ documents\ \cap Retrieved\ documents|}{|Retrieved\ documents|} \tag{6.1}$$

6.3.2 Recall

Recall is the fraction of the documents that are relevant to the query and successfully retrieved.

$$Recall = \frac{|Relevant\ documents\ \cap Retrieved\ documents|}{|Relevant\ documents|} \tag{6.2}$$

6.3.3 Entropy

The entropy provides a measure of the quality of clustering for flat clusters or the clusters at one level of the hierarchical clustering.

For each cluster, first, the class distribution is computed as P_{ij} i.e. the probability that a document in cluster j belongs to class i. Then, the entropy of each cluster j is calculated by summing over all classes as shown in equation 6.1.

$$E_j = -\sum_{i=1}^{m} P_{ij} log P_{ij} \tag{6.3}$$

Finally, overall entropy is calculated as the sum of each cluster's entropy weighted by the size of each cluster.

$$E = \sum_{j=1}^{m} \frac{n_j * E_j}{n} \tag{6.4}$$

Where n_j is the size of cluster j, m is the number of clusters, and n is the total number of documents.

6.3.4 F-Measure

The F-Measure is helpful to measure the effectiveness of hierarchical clustering or multi-level clustering. It combines the precision and recall ideas from the information retrieval [54].

We first compute the precision and recall for each given class, such as for cluster j and class i,

$$Precision(i, j) = \frac{n_{ij}}{n_j} \tag{6.5}$$

$$Recall(i, j) = \frac{n_{ij}}{n_i} \tag{6.6}$$

Where n_{ij} is the total documents for class i in cluster j, n_j is the total documents in cluster j, and n_i is the total documents with class i.

Then, the F-Measure for cluster j and class i is calculated as,

$$F(i,j) = \frac{2 * Precision(i,j) * Recall(i,j)}{Precision(i,j) + Recall(i,j)} \tag{6.7}$$

Overall F-Measure is calculated by taking the weighted average over all values of the F-Measure as in equation 6.6.

$$F = \sum_{i=1}^{m} \frac{n_i}{n} max[F(i,j)] \tag{6.8}$$

Where max is taken over all clusters at all levels, and n is the total number of documents.

6.3.5 Cosine Similarity

The Cosine Similarity between two document/sentence vectors is a measure that calculates the cosine of the angle between them. This metric is a measurement of orientation and not magnitude. It can be seen as a comparison between documents on a normalized space.

The cosine similarity equation is described as below,

$$cos(x,y) = \frac{x \cdot y}{||x|| \cdot ||y||} \tag{6.9}$$

6.3.6 ROUGE

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation [55]. It is essentially a set of metrics for evaluating automatic summarization of texts. It works by comparing an automatically produced summary with a human-produced reference summary.

Below is the list of ROUGE metrics used to evaluate the automatic summaries,

1. ROUGE-N: Overlap of N-grams between the automatic and reference summaries

[56].

2. ROUGE-1: Overlap of unigram between the automatic and reference summaries.

3. ROUGE-2: Overlap of bigrams between the automatic and reference summaries.

4. ROUGE-L: Longest Common Subsequence (LCS) based statistics [57]. LCS problem takes into account sentence level structure similarity and identifies the longest co-occurring in sequence n-grams.

CHAPTER 7: RESULTS AND DISCUSSION

In this chapter, we present and discuss the results of experiments, which are independently (or combinedly) performed for the document clustering and multi-document summarization.

7.1 Document Clustering

In all document clustering experiments, we use the Bisecting K-Means algorithm with I_2 criterion.

In the first experiment, each document in the 20 Newsgroups dataset is conceptualized. For every document, the document vector is generated by averaging the concept embeddings of the concepts that are present in the document. Then, the document clustering is performed on such document representation. The performance of such document clustering approach is very poor (Purity: 0.367, Entropy: 0.639) as shown in Figure 7.1. The reason for such bad performance is the loss of word ordering and semantics which happens due to averaging the document vectors.

20-way clustering: [I2=1.90e+04] [19997 of 19997], Entropy: 0.639, Purity: 0.367

cid	Size	ISim	ISdev	ESim	ESdev	Entpy	Purty	alt.	comp	comp	comp	comp	comp	misc	rec.	rec.	rec.	rec.	sci.	sci.	sci.	sci.	soc.	talk	talk	talk	talk
0	224	+0.864	+0.043	+0.773	+0.071	0.760	0.290	2	6	6	8	6	4	38	5	9	43	65	4	5	1	7	1	1	5	6	2
1	984	+0.928	+0.029	+0.870	+0.031	0.537	0.405	399	1	3	0	0	0	1	3	1	2	0	7	0	60	11	192	5	29	39	231
2	1029	+0.914	+0.031	+0.858	+0.036	0.374	0.581	144	0	0	0	0	0	0	0	0	0	0	0	0	1	2	598	10	13	18	243
3	682	+0.911	+0.033	+0.860	+0.038	0.185	0.886	15	1	0	0	0	0	0	1	0	1	1	0	0	2	0	9	7	604	31	10
4	1361	+0.927	+0.023	+0.879	+0.026	0.642	0.300	77	0	1	1	0	0	3	19	14	5	3	97	5	25	33	28	407	128	408	107
5	977	+0.896	+0.030	+0.852	+0.037	0.545	0.351	0	178	343	57	33	287	29	0	0	0	1	25	16	0	7	1	0	0	0	0
6	898	+0.897	+0.031	+0.856	+0.038	0.574	0.357	0	70	73	321	183	20	161	1	1	0	0	6	62	0	0	0	0	0	0	0
7	742	+0.906	+0.032	+0.868	+0.038	0.234	0.728	0	1	1	0	0	0	2	1	2	540	187	1	0	3	1	0	1	0	2	0
8	1642	+0.913	+0.026	+0.877	+0.028	0.732	0.189	4	283	117	145	199	299	45	23	12	6	3	139	310	23	27	0	1	1	3	2
9	648	+0.893	+0.036	+0.859	+0.041	0.164	0.912	6	0	0	2	0	0	1	1	9	2	0	0	5	591	3	1	11	0	8	8
10	1956	+0.923	+0.020	+0.889	+0.020	0.931	0.107	186	45	19	23	44	28	25	81	169	110	102	94	60	117	85	85	210	68	204	201
11	703	+0.903	+0.026	+0.871	+0.030	0.420	0.710	26	14	1	2	3	5	0	9	6	6	1	1	61	24	499	0	10	3	6	26
12	1105	+0.909	+0.030	+0.877	+0.032	0.768	0.374	7	85	45	34	52	41	56	58	9	13	3	413	72	33	110	7	12	13	35	7
13	1390	+0.903	+0.031	+0.874	+0.035	0.537	0.427	4	3	11	6	7	8	90	593	430	2	9	3	135	7	21	0	35	5	7	14
14	550	+0.875	+0.042	+0.847	+0.050	0.208	0.835	0	1	0	0	0	2	7	3	1	69	459	1	3	0	2	0	1	1	0	0
15	1157	+0.891	+0.029	+0.862	+0.033	0.841	0.252	4	65	66	107	126	81	292	60	73	33	19	34	84	17	50	9	7	11	9	10
16	1661	+0.901	+0.025	+0.875	+0.028	0.790	0.170	12	188	256	256	283	178	99	31	21	22	8	85	133	10	48	2	14	2	6	7
17	1152	+0.893	+0.028	+0.871	+0.031	0.877	0.189	71	18	13	4	14	13	37	39	47	45	18	58	22	54	54	46	218	94	187	100
18	1055	+0.865	+0.042	+0.851	+0.048	0.919	0.180	41	37	37	29	44	26	103	67	190	96	116	31	25	28	38	17	48	21	29	32
19	81	+0.591	+0.046	+0.583	+0.124	0.920	0.136	2	4	8	5	6	8	11	5	6	5	5	1	2	4	2	1	2	2	2	0

Figure 7.1: The performance of Document Clustering algorithm with AvgPatent2Vec model.

In the next experiment, each document in the 20 NewsGroups dataset is represented

with a paragraph vector. The benefit of using the paragraph vectors is to preserve the word ordering and its semantics. For generating these paragraph vectors, the Doc2Vec model is trained on the document collection using the Distributed Memory Model of Paragraph Vectors (PV-DM) algorithm. In addition, all the words with the frequency less than 5 are removed from the vocabulary. The context window size is kept to 8, and the noise is added to the input data by negative sampling. The Doc2Vec training is performed with 10 passes to produce the embedding with size equal to 300. Then, the document clustering is performed on these paragraph vectors. This approach improves the cluster quality approximately by 6%.

Next, the word vocabulary from Doc2Vec model is intersected with the pre-trained Word2Vec embedding [58]. Additionally, the document collection is shuffled after every pass during the Doc2Vec training. This approach significantly improves the document clustering performance (Purity: 0.617, Entropy: 0.417) as described in Figure 7.2.

20-way clustering: [I2=5.66e+03] [19997 of 19997], Entropy: 0.417, Purity: 0.617

cid	Size	ISim	ISdev	ESim	ESdev	Entpy	Purty	alt.	comp	comp	comp	comp	comp	misc	rec.	rec.	rec.	rec.	sci.	sci.	sci.	sci.	soc.	talk	talk	talk	talk
0	1029	+0.111	+0.027	+0.007	+0.006	0.152	0.918	4	6	1	0	1	2	2	3	3	37	945	1	6	5	3	0	0	1	5	4
1	1050	+0.114	+0.031	+0.012	+0.005	0.361	0.764	3	13	10	32	66	3	802	37	20	2	1	3	35	4	6	1	5	1	2	4
2	922	+0.104	+0.026	+0.007	+0.006	0.131	0.935	1	3	1	1	0	0	1	1	12	862	8	0	0	6	4	2	1	4	12	3
3	891	+0.088	+0.024	-0.001	+0.005	0.447	0.505	17	0	0	0	0	6	2	8	2	0	0	10	1	6	2	5	450	4	206	172
4	849	+0.085	+0.024	-0.001	+0.006	0.143	0.925	4	8	2	1	2	2	0	0	1	0	0	785	18	0	8	0	3	5	10	0
5	630	+0.082	+0.022	-0.001	+0.006	0.212	0.862	13	0	1	0	0	0	0	1	1	0	1	3	1	1	3	5	8	543	45	4
6	1153	+0.091	+0.024	+0.010	+0.005	0.487	0.526	3	64	607	262	104	42	12	0	0	0	2	13	34	2	2	2	0	1	1	2
7	512	+0.078	+0.028	-0.002	+0.006	0.425	0.668	47	0	5	0	0	0	0	4	1	0	0	6	1	3	6	18	12	342	52	15
8	1029	+0.087	+0.022	+0.007	+0.006	0.372	0.646	1	214	87	16	12	665	0	2	0	0	1	11	5	5	7	0	1	0	0	2
9	1394	+0.090	+0.022	+0.011	+0.005	0.479	0.440	3	80	52	489	613	15	60	3	3	0	0	6	63	3	3	0	0	0	0	1
10	1280	+0.070	+0.019	-0.004	+0.006	0.363	0.602	161	1	1	0	0	1	0	1	2	0	0	0	1	2	4	770	8	6	16	306
11	1396	+0.085	+0.025	+0.012	+0.007	0.789	0.349	16	487	121	72	46	159	31	38	24	36	11	39	104	67	58	38	12	8	11	18
12	1094	+0.067	+0.019	-0.002	+0.005	0.468	0.530	580	9	7	0	1	3	0	1	4	5	3	2	1	39	14	102	1	8	29	285
13	847	+0.068	+0.019	+0.001	+0.006	0.208	0.885	14	1	1	1	1	0	0	4	8	2	1	2	8	750	3	4	5	2	22	18
14	889	+0.073	+0.020	+0.009	+0.005	0.424	0.695	2	14	5	63	62	0	20	21	9	0	0	21	618	30	19	0	4	0	1	0
15	1696	+0.069	+0.018	+0.006	+0.005	0.361	0.474	9	3	4	5	12	2	17	767	804	2	0	0	34	5	5	2	8	2	8	7
16	885	+0.063	+0.018	+0.003	+0.005	0.216	0.878	22	14	0	0	2	1	2	5	3	0	0	1	6	16	777	2	2	1	14	17
17	796	+0.056	+0.015	-0.002	+0.005	0.542	0.545	31	2	0	0	0	0	0	31	29	1	0	61	10	2	14	10	434	21	121	29
18	993	+0.072	+0.024	+0.015	+0.009	0.974	0.095	37	79	85	57	77	94	41	58	47	49	25	28	52	42	32	19	35	38	43	55
19	662	+0.050	+0.016	+0.000	+0.005	0.551	0.607	32	2	10	1	1	5	10	15	27	4	2	8	2	12	30	17	11	13	402	58

Figure 7.2: The performance of Document Clustering algorithm with Doc2Vec model. Here, the Doc2Vec vocabulary is intersected with the pre-trained Word2Vec embedding.

In further document clustering experiments, the 20 NewsGroups document collection is first conceptualized with the pre-trained concept embedding model [38]. So, each document can be viewed as an n-gram bag-of-concepts (n <= 8). Then, the Doc2Vec model is created with the vocabulary of unique n-gram concepts and in-

tersected with the pre-trained concept embedding. For the Doc2Vec training, the similar setting is used as described in the previous experiment. Finally, it generates the document embeddings with dimension equal to 500. This approach boosts the document clustering performance by additional 7%.

20-way clustering: [I2=6.89e+03] [19997 of 19997], Entropy: 0.352, Purity: 0.697

cid	Size	ISim	ISdev	ESim	ESdev	Entpy	Purty	alt.	comp	comp	comp	comp	comp	misc	rec.	rec.	rec.	rec.	sci.	sci.	sci.	sci.	soc.	talk	talk	talk	talk
0	1013	+0.168	+0.033	+0.021	+0.009	0.108	0.947	2	1	1	0	0	2	1	1	3	22	959	0	1	3	1	3	3	2	7	1
1	962	+0.160	+0.033	+0.022	+0.009	0.070	0.967	0	0	0	0	0	0	1	0	0	930	6	0	1	5	1	0	5	2	9	2
2	350	+0.135	+0.032	+0.016	+0.007	0.171	0.880	0	15	1	2	1	0	0	0	0	0	0	308	21	1	0	0	0	0	0	1
3	956	+0.146	+0.030	+0.028	+0.008	0.463	0.538	0	14	41	514	193	2	86	0	2	0	0	2	95	1	1	1	0	0	2	2
4	1060	+0.141	+0.036	+0.025	+0.010	0.436	0.721	1	10	23	31	48	7	764	29	21	8	13	2	51	7	13	0	11	7	8	6
5	960	+0.134	+0.029	+0.020	+0.007	0.260	0.782	1	140	39	4	7	751	0	1	0	0	0	2	3	2	6	0	1	1	0	2
6	999	+0.122	+0.027	+0.017	+0.007	0.424	0.533	11	2	0	0	1	0	1	8	3	2	0	6	2	4	1	7	532	7	230	182
7	947	+0.129	+0.027	+0.026	+0.008	0.229	0.843	2	0	6	1	8	0	25	798	76	0	0	0	19	0	2	0	6	0	2	2
8	675	+0.117	+0.027	+0.014	+0.007	0.208	0.881	3	4	1	1	2	2	3	2	0	0	0	595	6	2	5	0	18	11	20	0
9	1096	+0.127	+0.028	+0.027	+0.007	0.457	0.593	1	129	650	139	67	55	14	0	0	0	1	3	29	1	2	1	0	0	4	0
10	1045	+0.125	+0.028	+0.030	+0.008	0.469	0.532	0	96	77	224	556	11	40	1	0	0	0	2	34	1	2	1	0	0	0	0
11	1048	+0.120	+0.028	+0.027	+0.007	0.282	0.823	14	1	3	0	5	3	8	82	862	2	3	1	5	10	9	1	9	17	8	5
12	863	+0.113	+0.026	+0.019	+0.008	0.143	0.924	5	0	0	0	1	0	0	3	2	0	0	1	4	797	7	5	3	1	20	14
13	1364	+0.102	+0.021	+0.013	+0.008	0.368	0.595	168	2	0	1	0	0	0	0	0	1	0	2	2	5	3	811	6	9	19	335
14	1401	+0.113	+0.028	+0.026	+0.012	0.744	0.399	13	559	133	49	47	159	28	18	13	30	13	46	92	56	63	29	10	21	7	15
15	1040	+0.096	+0.022	+0.011	+0.008	0.230	0.839	47	0	2	0	0	1	0	2	0	0	0	1	0	0	1	13	12	873	69	19
16	841	+0.109	+0.024	+0.027	+0.007	0.378	0.743	3	12	10	34	60	1	20	25	4	0	0	15	625	17	10	0	5	0	0	0
17	1276	+0.099	+0.022	+0.018	+0.008	0.473	0.529	675	9	8	0	1	3	0	0	1	5	3	2	2	62	13	110	6	18	33	325
18	910	+0.098	+0.023	+0.017	+0.007	0.144	0.924	11	6	1	0	2	2	1	0	1	0	0	0	3	14	841	0	1	3	17	7
19	1191	+0.073	+0.015	+0.016	+0.007	0.522	0.458	43	0	4	0	1	1	8	30	12	0	2	12	5	12	19	15	372	28	545	82

Figure 7.3: The performance of Document Clustering algorithm with Doc2Vec model. Here, the Doc2Vec vocabulary is intersected with the pre-trained concept embedding.

In the last experiment, the document preprocessing step additionally performs the lemmatization on unigrams. In this way, all different word forms with the same lemma are reduced to the same dictionary form and will not be treated differently during the Doc2Vec training. This preprocessing step further increases the document clustering performance approximately by 1%.

20-way clustering: [I2=6.29e+03] [19997 of 19997], Entropy: 0.344, Purity: 0.709

cid	Size	ISim	ISdev	ESim	ESdev	Entpy	Purty	alt.	comp	comp	comp	comp	comp	misc	rec.	rec.	rec.	rec.	sci.	sci.	sci.	sci.	soc.	talk	talk	talk	talk
0	1034	+0.143	+0.031	+0.009	+0.007	0.132	0.929	6	1	2	1	0	1	2	1	4	35	961	1	2	3	0	2	4	0	7	1
1	950	+0.134	+0.032	+0.010	+0.007	0.097	0.956	2	1	0	1	1	0	2	1	0	908	5	0	2	3	3	2	2	4	10	3
2	857	+0.133	+0.032	+0.016	+0.006	0.507	0.473	0	142	142	405	81	14	37	1	0	0	0	0	34	0	1	0	0	0	0	0
3	918	+0.116	+0.026	+0.009	+0.006	0.212	0.839	0	94	32	4	6	770	0	1	0	0	0	2	1	1	4	0	1	1	0	1
4	1132	+0.118	+0.034	+0.013	+0.007	0.482	0.686	3	20	29	43	54	11	777	20	19	12	16	7	56	5	19	1	16	10	9	5
5	366	+0.110	+0.031	+0.006	+0.006	0.235	0.833	1	20	1	4	4	2	0	0	0	0	0	305	26	0	0	0	0	1	0	2
6	644	+0.106	+0.027	+0.003	+0.006	0.162	0.905	1	1	0	1	1	0	0	1	0	0	0	583	8	0	3	0	16	8	20	1
7	989	+0.104	+0.026	+0.004	+0.006	0.428	0.528	18	0	0	0	0	1	1	7	2	0	1	6	1	4	1	6	522	13	222	184
8	942	+0.110	+0.027	+0.015	+0.006	0.414	0.644	1	52	607	155	35	41	11	0	0	0	1	5	27	2	3	0	0	0	2	0
9	981	+0.106	+0.025	+0.012	+0.006	0.233	0.851	2	4	0	1	8	0	39	835	51	0	1	1	22	1	3	0	3	1	6	3
10	1164	+0.110	+0.024	+0.017	+0.006	0.383	0.618	2	20	21	275	719	1	56	3	2	0	0	0	61	0	1	1	0	1	1	0
11	1010	+0.101	+0.025	+0.013	+0.006	0.206	0.883	13	0	2	2	3	1	5	47	892	2	0	2	4	6	5	0	8	6	8	4
12	1322	+0.086	+0.021	−0.001	+0.007	0.349	0.618	134	2	0	1	1	0	0	0	0	1	0	1	1	4	1	817	6	7	18	328
13	893	+0.088	+0.024	+0.006	+0.007	0.157	0.917	9	2	0	0	0	2	0	3	1	2	0	1	6	819	6	8	3	0	15	16
14	1065	+0.080	+0.020	−0.001	+0.007	0.240	0.832	45	0	2	0	0	1	0	3	0	0	0	1	1	0	2	15	13	886	79	17
15	1459	+0.092	+0.026	+0.014	+0.009	0.735	0.418	14	610	141	56	44	147	32	20	12	33	11	40	90	57	60	32	12	20	12	16
16	1256	+0.078	+0.020	+0.002	+0.006	0.435	0.557	700	6	6	1	0	4	0	0	0	3	3	0	0	50	10	103	5	15	25	325
17	876	+0.088	+0.023	+0.013	+0.006	0.373	0.744	1	14	10	50	42	0	27	27	5	0	0	16	652	24	5	0	3	0	0	0
18	935	+0.079	+0.021	+0.006	+0.006	0.165	0.912	11	11	1	0	0	3	6	0	2	1	0	1	3	12	853	0	2	1	16	12
19	1204	+0.054	+0.013	+0.003	+0.006	0.517	0.457	37	0	4	0	1	1	5	30	10	3	1	28	3	9	20	10	384	26	550	82

Figure 7.4: The performance of Document Clustering algorithm with Doc2Vec model. Here, the Doc2Vec vocabulary is intersected with the pre-trained concept embedding. Additionally, the preprocessing step performs lemmatization.

Table 7.1: An overview of the document clustering experiments.

	Document clustering experiment	Purity	Entropy
1	Averaging the concept vectors for concepts within a document	0.367	0.639
2	Bisecting K-Means on document embeddings obtained by the Doc2Vec training	0.425	0.646
3	Bisecting K-Means on document embeddings obtained by intersecting the Doc2Vec model with pre-trained Word2Vec embedding	0.617	0.417
4	Bisecting K-Means on document embeddings obtained by intersecting the Doc2Vec model with pre-trained concept embedding	0.697	0.352
5	Bisecting K-Means on document embeddings obtained by intersecting the Doc2Vec model with pre-trained concept embedding (+ lemmatization)	0.709	0.344

Additionally, the experiments are performed for the document embeddings with different dimensions. The comparison of these experiments is shown in Table 7.2. The document embedding with dimension 500 yields better clustering performance.

Table 7.2: The document clustering experiments with different dimensions.

	Document embedding dimension	Purity
1	500	70.9%
2	200	69.3%
3	150	68.9%

The best F-Measure for our document clustering experiment on 20 NewsGroups dataset is 79.34%. The 20 NewsGroups dataset has six high-level categories and 20

sub-categories. So, the F-Measure is calculated by first computing the cluster purity for high-level and sub-level clusters (High-level cluster purity: 0.8781 and Sub-level cluster purity: 0.7087).

Finally, we compared our document clustering approach with other state-of-the-art techniques [59, 60]. And our approach outperforms the existing techniques approximately by 6% as shown in Table 7.3.

Table 7.3: A comparison of different state-of-the-art document clustering techniques using F-Measure.

	Document clustering technique	Document representation	F-Measure
1	K-Means	TF-IDF	0.55
2	Hierarchical Agglomerative Clustering	TF-IDF	0.56
3	Expectation-Maximization with Mixture Model	TF-IDF	0.60
4	Principle Direction Divisive Partitioning	TF-IDF	0.66
5	Constructive-Competition Clustering	TF-IDF	0.69
6	Dataless Hierarchical Classification	Category Embedding [60]	0.709
7	Dataless Hierarchical Classification	TransE2 Embedding based on entities and relations [61]	0.710
8	Dataless Hierarchical Classification	Word Embedding [58]	0.717
9	Dataless Hierarchical Classification	Hierarchical Entity Embedding [62]	0.718
10	Dataless Hierarchical Classification	Hierarchical Category Embedding [60]	0.731
11	Bisecting K-Means	Document embedding obtained by intersecting the Doc2Vec model with pre-trained concept embedding	0.793

7.2 Multi-document Summarization

For evaluating multi-document summarization approach, we picked 10 different topics randomly from the Opinosis dataset. We computed Rouge-1, Rouge-2, and Cosine Similarity for the summaries generated by our approach as shown in Table 7.4. We also performed the same evaluation for the summaries produced by the LexRank algorithm as in Table 7.5. Finally, both techniques are compared by taking the mean over all topics (see Table 7.6).

Table 7.4: The summary evaluation for our approach (Rouge-1, Rouge-2, and Cosine Similarity).

	Topic	Rouge-1	Rouge-2	Cosine Similarity
1	Video ipod nano 8gb	0.50	0.09	0.45
2	Speed windows7	0.47	0.08	0.36
3	Eyesight-issues amazon kindle	0.47	0.18	0.38
4	Fonts amazon kindle	0.55	0.16	0.41
5	Battery-life netbook 1005ha	0.39	0.13	0.38
6	Quality toyota camry 2007	0.48	0.19	0.44
7	Accuracy garmin nuvi 255W	0.52	0.11	0.34
8	Screen netbook 1005ha	0.48	0.16	0.54
9	Comfort honda accord 2008	0.45	0.13	0.44
10	Interior toyota camry 2007	0.53	0.19	0.45

Table 7.5: The summary evaluation for LexRank (Rouge-1, Rouge-2, and Cosine Similarity).

	Topic	Rouge-1	Rouge-2	Cosine Similarity
1	Video ipod nano 8gb	0.41	0.11	0.45
2	Speed windows7	0.46	0.11	0.39
3	Eyesight-issues amazon kindle	0.40	0.17	0.45
4	Fonts amazon kindle	0.43	0.09	0.38
5	Battery-life netbook 1005ha	0.25	0.11	0.37
6	Quality toyota camry 2007	0.48	0.18	0.46
7	Accuracy garmin nuvi 255W	0.53	0.09	0.32
8	Screen netbook 1005ha	0.49	0.17	0.49
9	Comfort honda accord 2008	0.48	0.15	0.41
10	Interior toyota camry 2007	0.55	0.22	0.47

Table 7.6: The comparison between our multi-document summarization approach and LexRank (Rouge-1, Rouge-2, and Cosine Similarity).

	Algorithm	Rouge-1	Rouge-2	Cosine Similarity
1	Proposed Algorithm	0.4835	0.1417	0.4179
2	LexRank	0.4468	0.1401	0.4207

As per the comparison shown in Table 7.6, our multi-document summarization approach outperforms the well-known LexRank algorithm approximately by 4%.

Below is one of the summarization example generated for the Toyota Camry 2007 review,

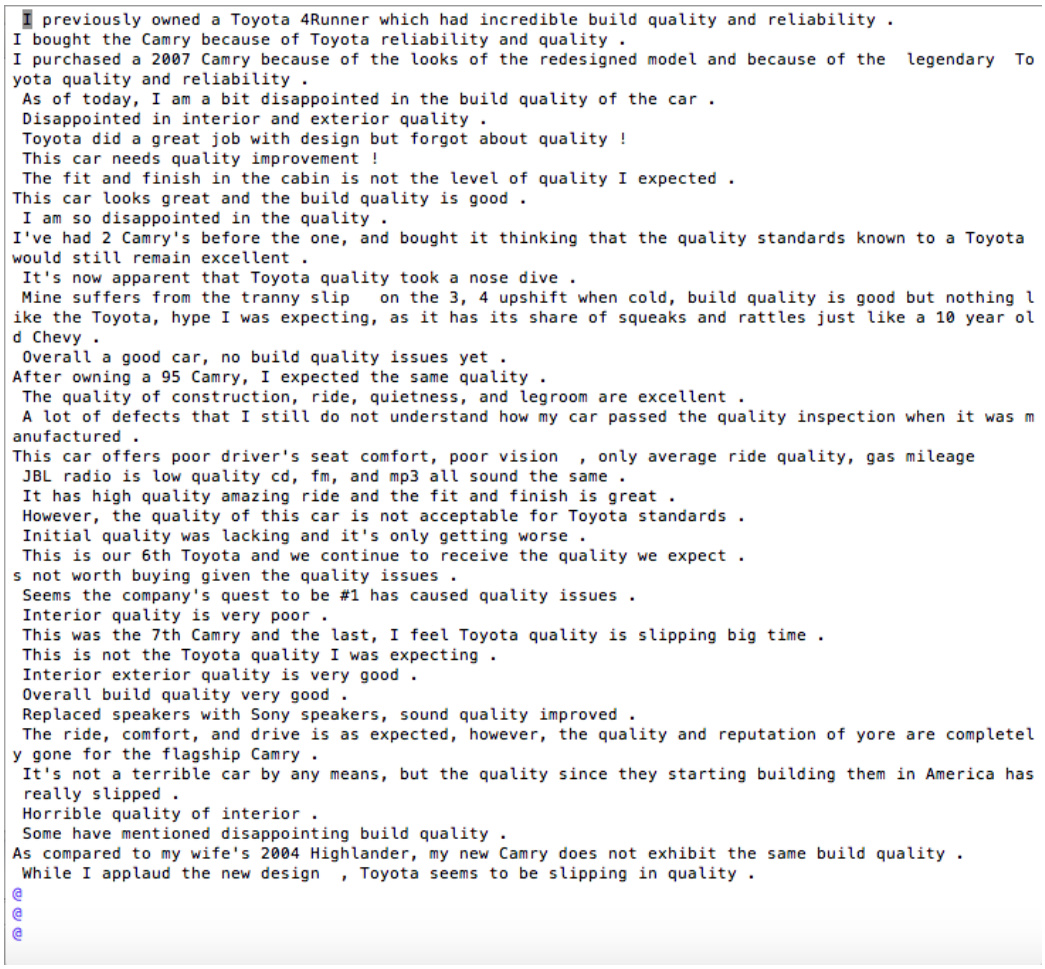

Figure 7.5: The original review of the Toyota Camry 2007.

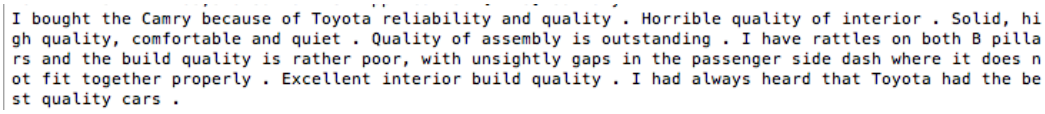

Figure 7.6: The summary of the Toyota Camry 2007 review.

7.3 Interactive Document Retrieval

In the Interactive Document Retrieval experiments, we used the BBC News Articles dataset. We performed an interactive search using proposed document clustering and multi-document summarization algorithms as shown in Figure 3.1.

In this experiment, we developed a command-line interactive search system. As a prerequisite, we first build an inverted index on BBC News articles using the tra-

ditional TF-IDF technique. Then, each article in BBC dataset is preprocessed and conceptualized. We train the PV-DM model on the conceptualized collection of BBC news articles, which we later use to generate the document embeddings. We also store the trained document embeddings in the database. Additionally, the Sentence2Vec model is trained on the sentence collection from the BBC News Articles.

Once the pre-requisites for an interactive search system are ready, we assess the system by providing it with the random search queries. First, the system loads the trained Sentence2Vec and Doc2Vec model in memory. Then, it prompts and asks the user to enter a search query as shown in Figure 7.7.

```
[cborse@cph-i2 InteractiveTextRetrieval]$ python InteractiveDocumentRetrievalApp.py
2017-11-23 17:02:52,884 summa.preprocessing.cleaner [INFO] 'pattern' package not found; tag filters are not
 available for English
2017-11-23 17:02:52,892 Interactive Document Retrieval [INFO] *****Interactive Document Retrieval*****
2017-11-23 17:02:52,892 Sentence2Vec [INFO] Loading Sentence2Vec model
2017-11-23 17:02:52,892 gensim.utils [INFO] loading Doc2Vec object from /users/cborse/final_demo/Interactiv
eTextRetrieval/output/models/sentence2vec.p2v
2017-11-23 17:02:53,197 gensim.utils [INFO] loading docvecs recursively from /users/cborse/final_demo/Inter
activeTextRetrieval/output/models/sentence2vec.p2v.docvecs.* with mmap=None
2017-11-23 17:02:53,198 gensim.utils [INFO] loading doctag_syn0 from /users/cborse/final_demo/InteractiveTe
xtRetrieval/output/models/sentence2vec.p2v.docvecs.doctag_syn0.npy with mmap=None
2017-11-23 17:02:53,222 gensim.utils [INFO] loading wv recursively from /users/cborse/final_demo/Interactiv
eTextRetrieval/output/models/sentence2vec.p2v.wv.* with mmap=None
2017-11-23 17:02:53,222 gensim.utils [INFO] setting ignored attribute syn0norm to None
2017-11-23 17:02:53,222 gensim.utils [INFO] setting ignored attribute cum_table to None
2017-11-23 17:02:53,222 gensim.utils [INFO] loaded /users/cborse/final_demo/InteractiveTextRetrieval/output
/models/sentence2vec.p2v
Enter a search query: mobile media player
```

Figure 7.7: The user prompt to enter a search query in the interactive search system.

Suppose, the user enters a search query "mobile media player". Then, the interactive search system first retrieves the relevant results using the traditional TF-IDF search and feeds it to the document clustering algorithm. The document clustering algorithm identifies the groups among the retrieved results and presents the user with the document clusters and their summaries. The below Figure 7.8 shows the interactive view. As shown in Figure 7.8, each document cluster has its unique aspect, such as business, sport, tech, etc.

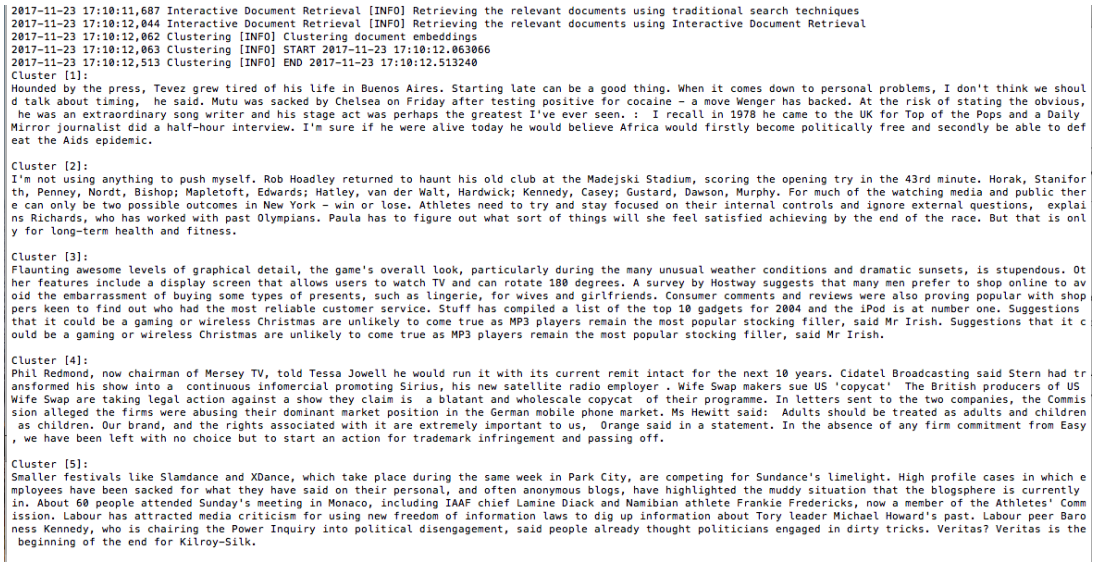

Figure 7.8: The interactive view of retrieved document clusters and their summaries.

Next, the user is asked for their choice of document clusters and the interactive search will be continued with the user-selected document clusters, as in Figure 7.9. Thus, the user can read the provided summary and pick the right cluster or clusters.

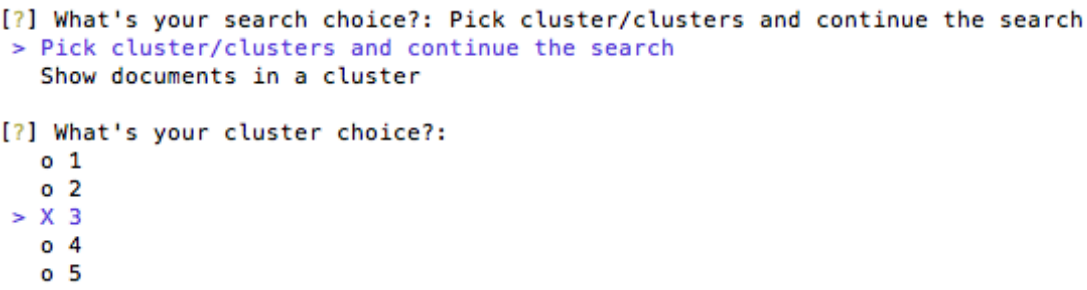

Figure 7.9: The interactive prompt for providing the choice of retrieved document clusters.

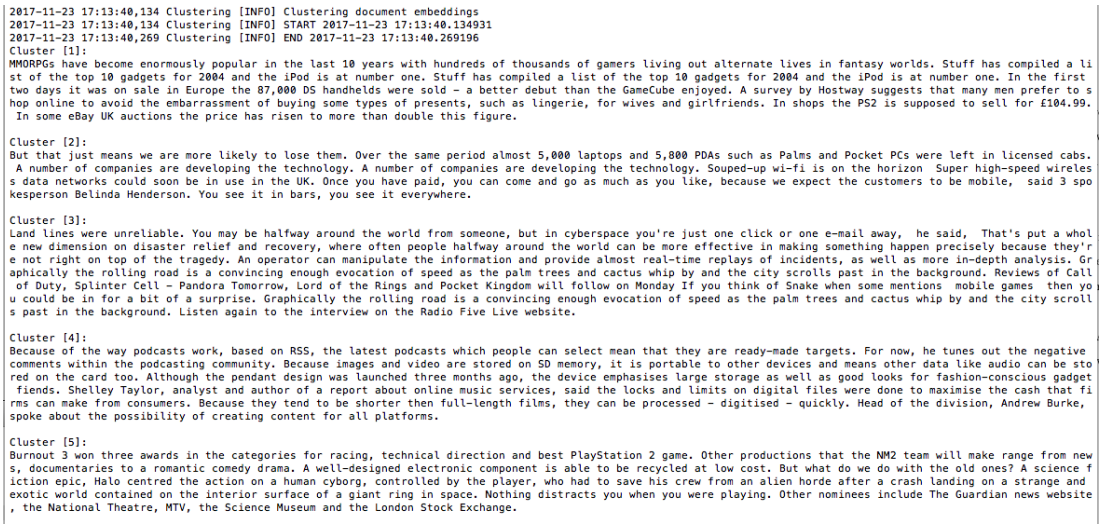

Figure 7.10: The interactive view of new document clusters and their summaries as per the user-selected document clusters.

The user optionally can view a collection of documents within each retrieved document cluster as in Figure 7.11 and pick the desired document as in Figure 7.12.

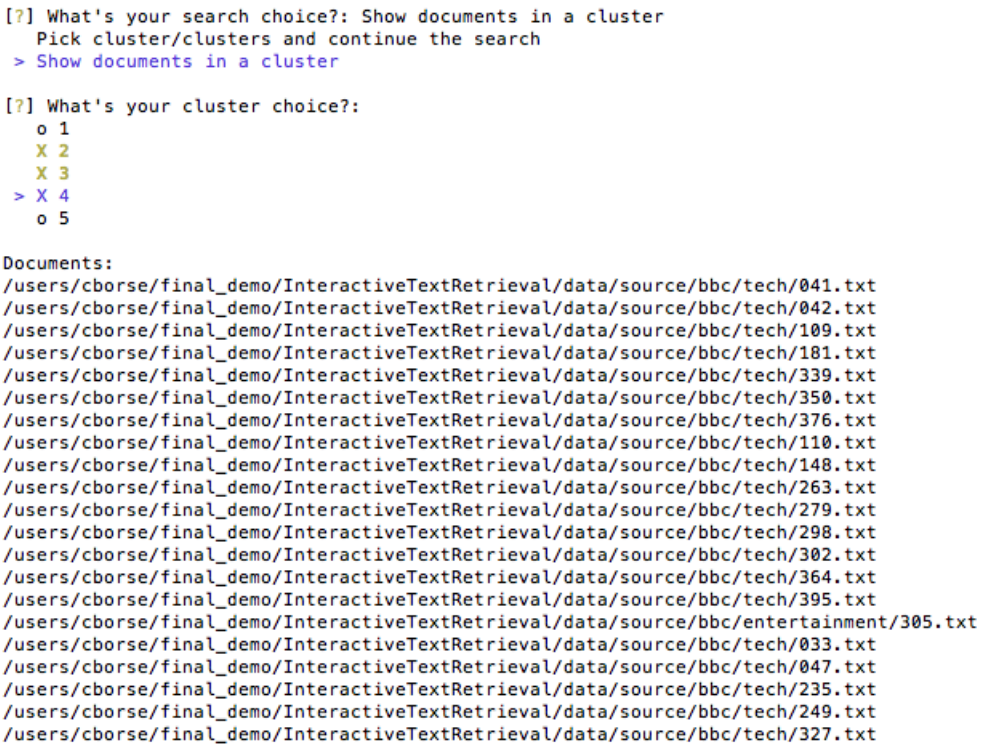

Figure 7.11: The interactive prompt for viewing the document list within retrieved clusters.

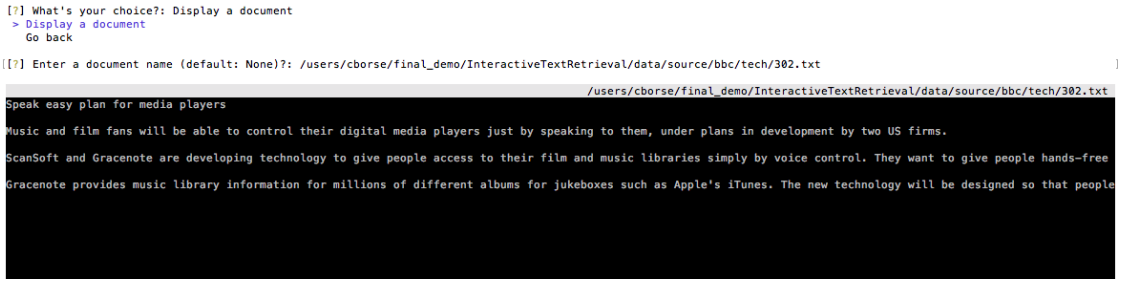

Figure 7.12: The interactive prompt for viewing a specific document within the retrieved clusters.

We performed a couple of experiments by running the ad-hoc search queries on the interactive search system. The experiments show that the interactive search system very accurately identifies the semantic groups of documents in the post-retrieved search results. Based on the user's interactions to provide a cluster choice, it ultimately helps the user to narrow down the search and to retrieve the desired document or documents.

We also performed few experiments with the ambiguous search queries. Our system successfully creates the separate document clusters as per the contexts in the retrieved documents. Such separation helps the user to filter out the unrelated documents and find the required results even if the ambiguous search query is provided.

CHAPTER 8: CONCLUSIONS

8.1 Conclusion

In this research, we have leveraged the Interactive Document Retrieval framework that finds the semantic groups of documents among the post-retrieved search results. The document clustering and the multi-document summarization are an integral part of the Interactive Document Retrieval framework. We have proposed a new approach for the document clustering and the multi-document summarization, and have improvised the framework to the next level.

We have proposed a new approach for the document clustering, which performs the clustering on document embeddings, known as paragraph vectors. These document embeddings are obtained by the Doc2Vec training on the conceptualized document collection. Our experiments on the 20 NewsGroups dataset show that the proposed document clustering approach outperforms the state-of-the-art techniques. The proposed algorithm improves the post-retrieved document clustering quality approximately by 6%.

We also proposed a new extraction-based multi-document summarization technique. Our technique extracts sentences from the document collection based on the highest importance scores computed using the Lexical Centrality principle. For power iteration, it builds the cosine similarity matrix using the sentence embeddings obtained with the PV-DM model. Additionally, the algorithm removes duplicate or most similar sentences from the multi-document summary by considering the similarity threshold 0.98. Thus, it generated the compact and diverse multi-document summary. This technique improves the multi-document summarization accuracy nearly by 4% as measured in Rouge-1 metrics.

In this way, our new document clustering and multi-document summarization approach improve the Interactive Document Retrieval framework. The improvised system helps the users to search the desired information easily and quickly. Thus, the proposed approach will add a new value to the interactive document retrieval and engage the users. Though our research focuses on the document retrieval, the proposed approach can be applied to other types of information retrieval systems such as the web search engine.

8.2 Future Scope

Our multi-document summarization technique generates the summary by performing a lot of computations run-time. So, it takes more time than the expectation to retrieve the desired results. The sentence embeddings generation, the cosine similarity matrix construction may be done offline. In this way, it will reduce the time required to generate the multi-document summary.

In future, the experiments can be performed to apply these interactive search techniques to the real-time search engine. Even though this research focuses on document retrieval, it will be interesting to see how it works for other retrieval domains.

REFERENCES

[1] T. Seymour, D. Frantsvog, and S. Kumar, "History of search engines," *International Journal of Management and Information Systems*, vol. 15, no. 4, p. 47, 2011.

[2] B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic, "Real life information retrieval: A study of user queries on the web," *SIGIR Forum*, vol. 32, no. 1, pp. 5–17, 1998.

[3] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1188–1196, 2014.

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

[5] M. A. Hearst and J. O. Pedersen, "Reexamining the cluster hypothesis: scatter/gather on retrieval results," in *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 76–84, ACM, 1996.

[6] M. J. Maña-López, M. De Buenaga, and J. M. Gómez-Hidalgo, "Multidocument summarization: An added value to clustering in interactive retrieval," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 2, pp. 215–241, 2004.

[7] G. Salton, "Automatic information organization and retrieval," 1968.

[8] O. Zamir and O. Etzioni, "Web document clustering: A feasibility demonstration," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 46–54, ACM, 1998.

[9] M. Carey, F. Kriwaczek, and S. M. Ruger, "A visualization interface for document searching and browsing," in *Proceedings of CIKM 2000 Workshop on New Paradigms in Information Visualization and Manipulation*, pp. 24–28, 2000.

[10] A. Leuski, "Evaluating document clustering for interactive information retrieval," in *Proceedings of the tenth international conference on Information and knowledge management*, pp. 33–40, ACM, 2001.

[11] M. Wu, M. Fuller, and R. Wilkinson, "Using clustering and classification approaches in interactive retrieval," *Information Processing & Management*, vol. 37, no. 3, pp. 459–484, 2001.

[12] Y. Wang and M. Kitsuregawa, "Evaluating contents-link coupled web page clustering for web search results," in *Proceedings of the eleventh international conference on Information and knowledge management*, pp. 499–506, ACM, 2002.

[13] S. Koshman, A. Spink, and B. J. Jansen, "Web searching on the vivisimo search engine," *Journal of the Association for Information Science and Technology*, vol. 57, no. 14, pp. 1875–1887, 2006.

[14] A. Tagarelli and G. Karypis, "A segment-based approach to clustering multi-topic documents," *Knowledge and information systems*, vol. 34, no. 3, pp. 563–595, 2013.

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

[16] M.-Y. Kan, K. R. McKeown, and J. L. Klavans, "Domain-specific informative and indicative summarization for information retrieval," in *Proc. of the Document Understanding Conference (DUC)*, pp. 19–26, 2001.

[17] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *Journal of Artificial Intelligence Research*, vol. 22, pp. 457–479, 2004.

[18] X. Wan, J. Yang, and J. Xiao, "Manifold-ranking based topic-focused multi-document summarization.," in *IJCAI*, vol. 7, pp. 2903–2908, 2007.

[19] K. Knight and D. Marcu, "Statistics-based summarization-step one: Sentence compression," *AAAI/IAAI*, vol. 2000, pp. 703–710, 2000.

[20] C.-Y. Lin, "Improving summarization performance by sentence compression: a pilot study," in *Proceedings of the sixth international workshop on Information retrieval with Asian languages-Volume 11*, pp. 1–8, Association for Computational Linguistics, 2003.

[21] D. M. Zajic, B. Dorr, J. Lin, and R. Schwartz, "Sentence compression as a component of a multi-document summarization system," in *Proceedings of the 2006 document understanding workshop, New York*, 2006.

[22] S. Harabagiu and F. Lacatusu, "Using topic themes for multi-document summarization," *ACM Transactions on Information Systems (TOIS)*, vol. 28, no. 3, p. 13, 2010.

[23] P. Li, L. Bing, W. Lam, H. Li, and Y. Liao, "Reader-aware multi-document summarization via sparse coding.," in *IJCAI*, pp. 1270–1276, 2015.

[24] J. C. K. Cheung and G. Penn, "Towards robust abstractive multi-document summarization: A caseframe analysis of centrality and domain.," in *ACL (1)*, pp. 1233–1242, 2013.

[25] Z. Cao, F. Wei, L. Dong, S. Li, and M. Zhou, "Ranking with recursive neural networks and its application to multi-document summarization.," in *AAAI*, pp. 2153–2159, 2015.

[26] "Information retrieval," Oct 2017.

[27] C. D. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval: Cambridge: 2008."

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

[29] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford InfoLab, 1999.

[30] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual web search engine," *Computer networks*, vol. 56, no. 18, pp. 3825–3833, 2012.

[31] P. R. Krishnaiah and L. N. Kanal, "Classification, pattern recognition, and reduction of dimensionality, volume 2 of handbook of statistics," *North4Holland Amsterdam*, 1982.

[32] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[33] L. Rokach and O. Maimon, "Clustering methods," in *Data mining and knowledge discovery handbook*, pp. 321–352, Springer, 2005.

[34] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[35] A. Spink, H. C. Ozmutlu, and S. Ozmutlu, "Multitasking information seeking and searching processes," *Journal of the Association for Information Science and Technology*, vol. 53, no. 8, pp. 639–652, 2002.

[36] M. Gao, L. Yuan, and W. T. Fu, "An interactive retrieval framework for online health information," in *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 303–303, Oct 2016.

[37] G. Zenz, X. Zhou, E. Minack, W. Siberski, and W. Nejdl, "From keywords to semantic queries - incremental query construction on the semantic web," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 3, pp. 166–176, 2009.

[38] W. Shalaby and W. Zadrozny, "Learning concept embeddings for efficient bag-of-concepts densification," *arXiv preprint arXiv:1702.03342*, 2017.

[39] M. Steinbach, G. Karypis, V. Kumar, *et al.*, "A comparison of document clustering techniques," in *KDD workshop on text mining*, vol. 400, pp. 525–526, Boston, 2000.

[40] S. Vijayarani, M. J. Ilamathi, and M. Nithya, "Preprocessing techniques for text mining-an overview," *International Journal of Computer Science & Communication Networks*, vol. 5, no. 1, pp. 7–16, 2015.

[41] E. M. Rasmussen, "Clustering algorithms.," *Information retrieval: data structures & algorithms*, vol. 419, p. 442, 1992.

[42] G. Karypis, "Cluto-a clustering toolkit," tech. rep., MINNESOTA UNIV MINNEAPOLIS DEPT OF COMPUTER SCIENCE, 2002.

[43] Y. Zhao and G. Karypis, "Criterion functions for document clustering: Experiments and analysis," tech. rep., Technical report, 2001.

[44] 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*, pp. 335–336, ACM, 1998.

[45] X. Zhu, A. B. Goldberg, J. Van Gael, and D. Andrzejewski, "Improving diversity in ranking using absorbing random walks.," in *HLT-NAACL*, pp. 97–104, 2007.

[46] D. R. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Celebi, S. Dimitrov, E. Drabek, A. Hakim, W. Lam, D. Liu, *et al.*, "Mead-a platform for multidocument multilingual text summarization.," in *LREC*, 2004.

[47] J. Clarke and M. Lapata, "Global inference for sentence compression: An integer linear programming approach," *Journal of Artificial Intelligence Research*, vol. 31, pp. 399–429, 2008.

[48] C. Smith and A. Jönsson, "Automatic summarization as means of simplifying texts, an evaluation for swedish," 2011.

[49] K. Lang, "Newsweeder: Learning to filter netnews," in *Proceedings of the 12th international conference on machine learning*, vol. 10, pp. 331–339, 1995.

[50] K. Ganesan, C. Zhai, and J. Han, "Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions," in *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 340–348, Association for Computational Linguistics, 2010.

[51] D. Greene and P. Cunningham, "Practical solutions to the problem of diagonal dominance in kernel document clustering," in *Proc. 23rd International Conference on Machine learning (ICML'06)*, pp. 377–384, ACM Press, 2006.

[52] C. E. Shannon, "A mathematical theory of communication, part i, part ii," *Bell Syst. Tech. J.*, vol. 27, pp. 623–656, 1948.

[53] B. Larsen and C. Aone, "Fast and effective text mining using linear-time document clustering," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 16–22, ACM, 1999.

[54] G. Kowalski, "Information retrieval systems: theory and implementation," *Computers & Mathematics with Applications*, vol. 35, no. 5, pp. 133–133, 1998.

[55] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out: Proceedings of the ACL-04 workshop*, vol. 8, Barcelona, Spain, 2004.

[56] C.-Y. Lin and E. Hovy, "Automatic evaluation of summaries using n-gram co-occurrence statistics," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pp. 71–78, Association for Computational Linguistics, 2003.

[57] C.-Y. Lin and F. J. Och, "Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics," in *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, p. 605, Association for Computational Linguistics, 2004.

[58] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[59] Y.-W. Seo and K. Sycara, "Text clustering for topic detection," tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 2004.

[60] Y. Li, R. Zheng, T. Tian, Z. Hu, R. Iyer, and K. Sycara, "Joint embedding of hierarchical categories and entities for concept categorization and dataless classification," *arXiv preprint arXiv:1607.07956*, 2016.

[61] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, pp. 2787–2795, 2013.

[62] Z. Hu, P. Huang, Y. Deng, Y. Gao, and E. P. Xing, "Entity hierarchy embedding.," in *ACL (1)*, pp. 1292–1300, 2015.

APPENDIX A: HIGH RESOLUTION FIGURES

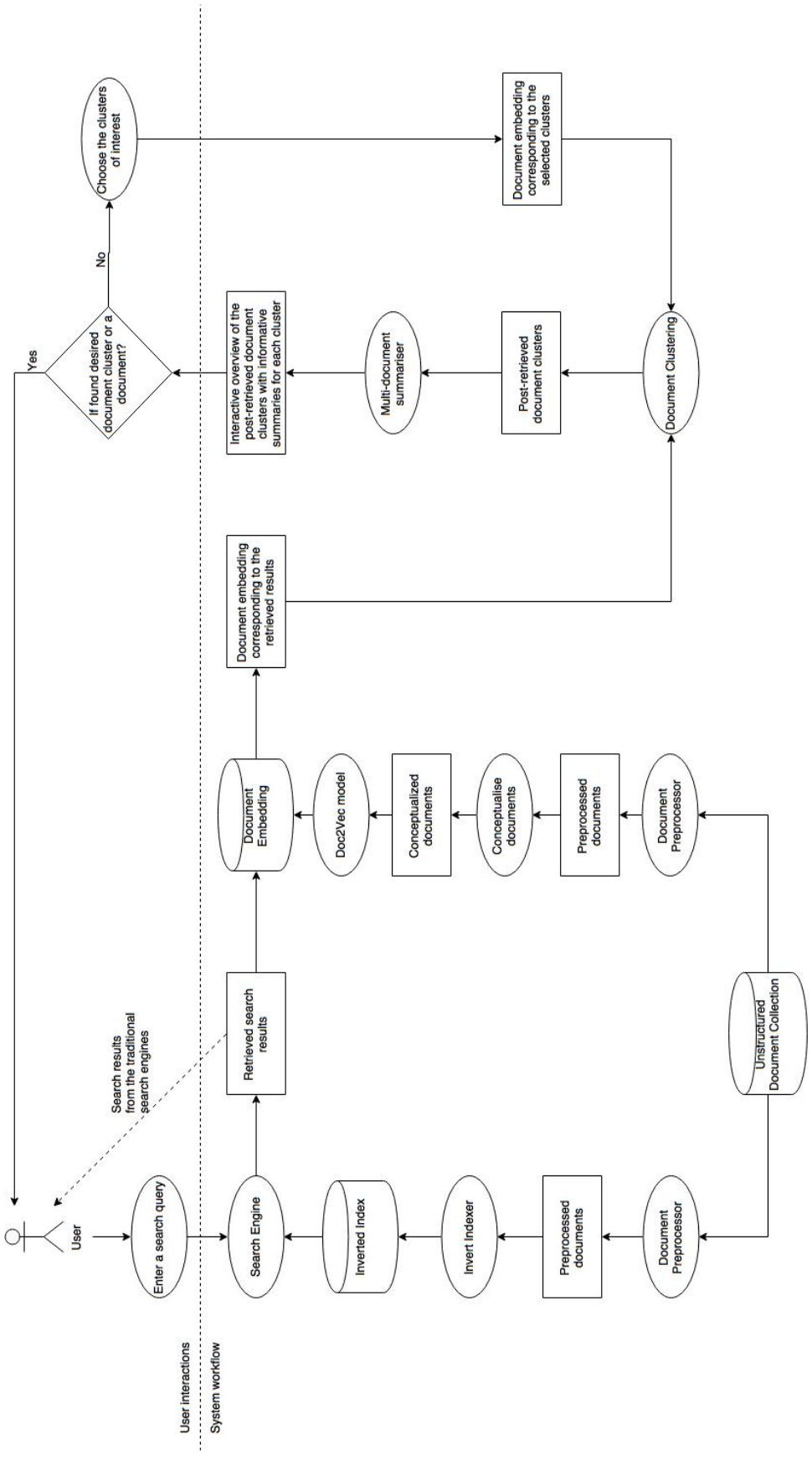

Figure A.1: An Interactive Document Retrieval framework.

20-way clustering: [I2=1.90e+04] [19997 of 19997], Entropy: 0.639, Purity: 0.367

cid	Size	ISim	ISdev	ESim	ESdev	Entpy	Purty	alt.	comp	comp	comp	comp	comp	misc	rec.	rec.	rec.	rec.	sci.	sci.	sci.	sci.	soc.	talk	talk	talk	talk
0	224	+0.864	+0.043	+0.773	+0.071	0.760	0.290	2	6	6	8	6	4	38	5	9	43	65	4	5	1	7	1	1	5	6	2
1	984	+0.928	+0.029	+0.870	+0.031	0.537	0.405	399	1	3	0	0	0	1	3	1	2	0	7	0	60	11	192	5	29	39	231
2	1029	+0.914	+0.031	+0.858	+0.036	0.374	0.581	144	0	0	0	0	0	0	0	0	0	0	0	0	1	2	598	10	13	18	243
3	682	+0.911	+0.033	+0.860	+0.038	0.185	0.886	15	1	0	0	0	0	0	1	0	1	1	0	0	2	0	9	7	604	31	10
4	1361	+0.927	+0.023	+0.879	+0.026	0.642	0.300	77	0	1	1	0	0	3	19	14	5	3	97	5	25	33	28	407	128	408	107
5	977	+0.896	+0.030	+0.852	+0.037	0.545	0.351	0	178	343	57	33	287	29	0	0	0	1	25	16	0	7	1	0	0	0	0
6	898	+0.897	+0.031	+0.856	+0.038	0.574	0.357	0	70	73	321	183	20	161	1	1	0	0	6	62	0	0	0	0	0	0	0
7	742	+0.906	+0.032	+0.868	+0.038	0.234	0.728	0	1	1	0	0	0	2	1	2	540	187	1	0	3	1	0	1	0	2	0
8	1642	+0.913	+0.026	+0.877	+0.028	0.732	0.189	4	283	117	145	199	299	45	23	12	6	3	139	310	23	27	0	1	1	3	2
9	648	+0.893	+0.036	+0.859	+0.041	0.164	0.912	6	0	0	2	0	0	1	1	9	2	0	0	5	591	3	1	11	0	8	8
10	1956	+0.923	+0.020	+0.889	+0.020	0.931	0.107	186	45	19	23	44	28	25	81	169	110	102	94	60	117	85	85	210	68	204	201
11	703	+0.903	+0.026	+0.871	+0.030	0.420	0.710	26	14	1	2	3	5	0	9	6	6	1	1	61	24	499	0	10	3	6	26
12	1105	+0.909	+0.030	+0.877	+0.032	0.768	0.374	7	85	45	34	52	41	56	58	9	13	3	413	72	33	110	7	12	13	35	7
13	1390	+0.903	+0.031	+0.874	+0.035	0.537	0.427	4	3	11	6	7	8	90	593	430	2	9	3	135	7	21	0	35	5	7	14
14	550	+0.875	+0.042	+0.847	+0.050	0.208	0.835	0	1	0	0	0	2	7	3	1	69	459	1	3	0	2	0	1	1	0	0
15	1157	+0.891	+0.029	+0.862	+0.033	0.841	0.252	4	65	66	107	126	81	292	60	73	33	19	34	84	17	50	9	7	11	9	10
16	1661	+0.901	+0.025	+0.875	+0.028	0.790	0.170	12	188	256	256	283	178	99	31	21	22	8	85	133	10	48	2	14	2	6	7
17	1152	+0.893	+0.028	+0.871	+0.031	0.877	0.189	71	18	13	4	14	13	37	39	47	45	18	58	22	54	54	46	218	94	187	100
18	1055	+0.865	+0.042	+0.851	+0.048	0.919	0.180	41	37	37	29	44	26	103	67	190	96	116	31	25	28	38	17	48	21	29	32
19	81	+0.591	+0.046	+0.583	+0.124	0.920	0.136	2	4	8	5	6	8	11	5	6	5	5	1	2	4	2	1	2	2	2	0

Figure A.2: The performance of Document Clustering algorithm with AvgPatent2Vec model.

20-way clustering: [I2=5.66e+03] [19997 of 19997], Entropy: 0.417, Purity: 0.617

cid	Size	ISim	ISdev	ESim	ESdev	Entpy	Purty	alt.	comp	comp	comp	comp	comp	misc	rec.	rec.	rec.	rec.	sci.	sci.	sci.	sci.	soc.	talk	talk	talk	talk
0	1029	+0.111	+0.027	+0.007	+0.006	0.152	0.918	4	6	1	0	1	2	2	3	3	37	945	1	6	5	3	0	0	1	5	4
1	1050	+0.114	+0.031	+0.012	+0.005	0.361	0.764	3	13	10	32	66	3	802	37	20	2	1	3	35	4	6	1	5	1	2	4
2	922	+0.104	+0.026	+0.007	+0.006	0.131	0.935	1	3	1	1	0	0	1	1	12	862	8	0	0	6	4	2	1	4	12	3
3	891	+0.088	+0.024	-0.001	+0.005	0.447	0.505	17	0	0	0	0	6	2	8	2	0	0	10	1	6	2	5	450	4	206	172
4	849	+0.085	+0.024	-0.001	+0.006	0.143	0.925	4	8	2	1	2	2	0	0	1	0	0	785	18	0	8	0	3	5	10	0
5	630	+0.082	+0.022	-0.001	+0.006	0.212	0.862	13	0	1	0	0	0	0	1	1	0	1	3	1	1	3	5	8	543	45	4
6	1153	+0.091	+0.024	+0.010	+0.005	0.487	0.526	3	64	607	262	104	42	12	0	0	0	2	13	34	2	2	2	0	1	1	2
7	512	+0.078	+0.028	-0.002	+0.006	0.425	0.668	47	0	5	0	0	0	0	4	1	0	0	6	1	3	6	18	12	342	52	15
8	1029	+0.087	+0.022	+0.007	+0.006	0.372	0.646	1	214	87	16	12	665	0	2	0	0	1	11	5	5	7	0	1	0	0	2
9	1394	+0.090	+0.022	+0.011	+0.005	0.479	0.440	3	80	52	489	613	15	60	3	3	0	0	6	63	3	3	0	0	0	0	1
10	1280	+0.070	+0.019	-0.004	+0.006	0.363	0.602	161	1	1	0	0	1	0	1	2	0	0	0	1	2	4	770	8	6	16	306
11	1396	+0.085	+0.025	+0.012	+0.007	0.789	0.349	16	487	121	72	46	159	31	38	24	36	11	39	104	67	58	38	12	8	11	18
12	1094	+0.067	+0.019	-0.002	+0.005	0.468	0.530	580	9	7	0	1	3	0	1	4	5	3	2	1	39	14	102	1	8	29	285
13	847	+0.068	+0.019	+0.001	+0.006	0.208	0.885	14	1	1	1	1	0	0	4	8	2	1	2	8	750	3	4	5	2	22	18
14	889	+0.073	+0.020	+0.009	+0.005	0.424	0.695	2	14	5	63	62	0	20	21	9	0	0	21	618	30	19	0	4	0	1	0
15	1696	+0.069	+0.018	+0.006	+0.005	0.361	0.474	9	3	4	5	12	2	17	767	804	2	0	0	34	5	5	2	8	2	8	7
16	885	+0.063	+0.018	+0.003	+0.005	0.216	0.878	22	14	0	0	2	1	2	5	3	0	0	1	6	16	777	2	2	1	14	17
17	796	+0.056	+0.015	-0.002	+0.005	0.542	0.545	31	2	0	0	0	0	0	31	29	1	0	61	10	2	14	10	434	21	121	29
18	993	+0.072	+0.024	+0.015	+0.009	0.974	0.095	37	79	85	57	77	94	41	58	47	49	25	28	52	42	32	19	35	38	43	55
19	662	+0.050	+0.016	+0.000	+0.005	0.551	0.607	32	2	10	1	1	5	10	15	27	4	2	8	2	12	30	17	11	13	402	58

Figure A.3: The performance of Document Clustering algorithm with Doc2Vec model. Here, the Doc2Vec vocabulary is intersected with the pre-trained Word2Vec embedding.

20-way clustering: [I2=6.89e+03] [19997 of 19997], Entropy: 0.352, Purity: 0.697

cid	Size	ISim	ISdev	ESim	ESdev	Entpy	Purty	alt.	comp	comp	comp	comp	comp	misc	rec.	rec.	rec.	rec.	sci.	sci.	sci.	sci.	soc.	talk	talk	talk	talk
0	1013	+0.168	+0.033	+0.021	+0.009	0.108	0.947	2	1	1	0	0	2	1	1	3	22	959	0	1	3	1	3	3	2	7	1
1	962	+0.160	+0.033	+0.022	+0.009	0.070	0.967	0	0	0	0	0	0	1	0	0	930	6	0	1	5	1	0	5	2	9	2
2	350	+0.135	+0.032	+0.016	+0.007	0.171	0.880	0	15	1	2	1	0	0	0	0	0	0	308	21	1	0	0	0	0	0	1
3	956	+0.146	+0.030	+0.028	+0.008	0.463	0.538	0	14	41	514	193	2	86	0	2	0	0	2	95	1	1	1	0	0	2	2
4	1060	+0.141	+0.036	+0.025	+0.010	0.436	0.721	1	10	23	31	48	7	764	29	21	8	13	2	51	7	13	0	11	7	8	6
5	960	+0.134	+0.029	+0.020	+0.007	0.260	0.782	1	140	39	4	7	751	0	1	0	0	0	2	3	2	6	0	1	1	0	2
6	999	+0.122	+0.027	+0.017	+0.007	0.424	0.533	11	2	0	0	1	0	1	8	3	2	0	6	2	4	1	7	532	7	230	182
7	947	+0.129	+0.027	+0.026	+0.008	0.229	0.843	2	0	6	1	8	0	25	798	76	0	0	0	19	0	2	0	6	0	2	2
8	675	+0.117	+0.027	+0.014	+0.007	0.208	0.881	3	4	1	1	2	2	3	2	0	0	0	595	6	2	5	0	18	11	20	0
9	1096	+0.127	+0.028	+0.027	+0.007	0.457	0.593	1	129	650	139	67	55	14	0	0	0	1	3	29	1	2	1	0	0	4	0
10	1045	+0.125	+0.028	+0.030	+0.008	0.469	0.532	0	96	77	224	556	11	40	1	0	0	0	2	34	1	2	1	0	0	0	0
11	1048	+0.120	+0.028	+0.027	+0.007	0.282	0.823	14	1	3	0	5	3	8	82	862	2	3	1	5	10	9	1	9	17	8	5
12	863	+0.113	+0.026	+0.019	+0.008	0.143	0.924	5	0	0	0	1	0	0	3	2	0	0	1	4	797	7	5	3	1	20	14
13	1364	+0.102	+0.021	+0.013	+0.008	0.368	0.595	168	2	0	1	0	0	0	0	0	1	0	2	2	5	3	811	6	9	19	335
14	1401	+0.113	+0.028	+0.026	+0.012	0.744	0.399	13	559	133	49	47	159	28	18	13	30	13	46	92	56	63	29	10	21	7	15
15	1040	+0.096	+0.022	+0.011	+0.008	0.230	0.839	47	0	2	0	0	1	0	2	0	0	0	1	0	0	1	13	12	873	69	19
16	841	+0.109	+0.024	+0.027	+0.007	0.378	0.743	3	12	10	34	60	1	20	25	4	0	0	15	625	17	10	0	5	0	0	0
17	1276	+0.099	+0.022	+0.018	+0.008	0.473	0.529	675	9	8	0	1	3	0	0	1	5	3	2	2	62	13	110	6	18	33	325
18	910	+0.098	+0.023	+0.017	+0.007	0.144	0.924	11	6	1	0	2	2	1	0	1	0	0	0	3	14	841	0	1	3	17	7
19	1191	+0.073	+0.015	+0.016	+0.007	0.522	0.458	43	0	4	0	1	1	8	30	12	0	2	12	5	12	19	15	372	28	545	82

Figure A.4: The performance of Document Clustering algorithm with Doc2Vec model. Here, the Doc2Vec vocabulary is intersected with the pre-trained concept embedding.

20-way clustering: [I2=6.29e+03] [19997 of 19997], Entropy: 0.344, Purity: 0.709

cid	Size	ISim	ISdev	ESim	ESdev	Entpy	Purty	alt.	comp	comp	comp	comp	comp	misc	rec.	rec.	rec.	rec.	sci.	sci.	sci.	sci.	soc.	talk	talk	talk	talk
0	1034	+0.143	+0.031	+0.009	+0.007	0.132	0.929	6	1	2	1	0	1	2	1	4	35	961	1	2	3	0	2	4	0	7	1
1	950	+0.134	+0.032	+0.010	+0.007	0.097	0.956	2	1	0	1	1	0	2	1	0	908	5	0	2	3	3	2	2	4	10	3
2	857	+0.133	+0.032	+0.016	+0.006	0.507	0.473	0	142	142	405	81	14	37	1	0	0	0	0	34	0	1	0	0	0	0	0
3	918	+0.116	+0.026	+0.009	+0.006	0.212	0.839	0	94	32	4	6	770	0	1	0	0	0	2	1	1	4	0	1	1	0	1
4	1132	+0.118	+0.034	+0.013	+0.007	0.482	0.686	3	20	29	43	54	11	777	20	19	12	16	7	56	5	19	1	16	10	9	5
5	366	+0.110	+0.031	+0.006	+0.006	0.235	0.833	1	20	1	4	4	2	0	0	0	0	0	305	26	0	0	0	0	1	0	2
6	644	+0.106	+0.027	+0.003	+0.006	0.162	0.905	1	1	0	1	1	0	0	1	0	0	0	583	8	0	3	0	16	8	20	1
7	989	+0.104	+0.026	+0.004	+0.006	0.428	0.528	18	0	0	0	0	1	1	7	2	0	1	6	1	4	1	6	522	13	222	184
8	942	+0.110	+0.027	+0.015	+0.006	0.414	0.644	1	52	607	155	35	41	11	0	0	0	1	5	27	2	3	0	0	0	2	0
9	981	+0.106	+0.025	+0.012	+0.006	0.233	0.851	2	4	0	1	8	0	39	835	51	0	1	1	22	1	3	0	3	1	6	3
10	1164	+0.110	+0.024	+0.017	+0.006	0.383	0.618	2	20	21	275	719	1	56	3	2	0	0	0	61	0	1	1	0	1	1	0
11	1010	+0.101	+0.025	+0.013	+0.006	0.206	0.883	13	0	2	2	3	1	5	47	892	2	0	2	4	6	5	0	8	6	8	4
12	1322	+0.086	+0.021	-0.001	+0.007	0.349	0.618	134	2	0	1	1	0	0	0	0	1	0	1	1	4	1	817	6	7	18	328
13	893	+0.088	+0.024	+0.006	+0.007	0.157	0.917	9	2	0	0	0	2	0	3	1	2	0	1	6	819	6	8	3	0	15	16
14	1065	+0.080	+0.020	-0.001	+0.007	0.240	0.832	45	0	2	0	0	1	0	3	0	0	0	1	1	0	2	15	13	886	79	17
15	1459	+0.092	+0.026	+0.014	+0.009	0.735	0.418	14	610	141	56	44	147	32	20	12	33	11	40	90	57	60	32	12	20	12	16
16	1256	+0.078	+0.020	+0.002	+0.006	0.435	0.557	700	6	6	1	0	4	0	0	0	3	3	0	0	50	10	103	5	15	25	325
17	876	+0.088	+0.023	+0.013	+0.006	0.373	0.744	1	14	10	50	42	0	27	27	5	0	0	16	652	24	5	0	3	0	0	0
18	935	+0.079	+0.021	+0.006	+0.006	0.165	0.912	11	11	1	0	0	3	6	0	2	1	0	1	3	12	853	0	2	1	16	12
19	1204	+0.054	+0.013	+0.003	+0.006	0.517	0.457	37	0	4	0	1	1	5	30	10	3	1	28	3	9	20	10	384	26	550	82

Figure A.5: The performance of Document Clustering algorithm with Doc2Vec model. Here, the Doc2Vec vocabulary is intersected with the pre-trained concept embedding. Additionally, the preprocessing step performs lemmatization.

VITA

Chetan Borse was born and raised in Loni, India. Before attending the University of North Carolina at Charlotte, he attended the University of Pune, India, where he earned a Bachelor of Engineering, with Highest Distinction, in 2012. From 2012 to 2015, he worked as a Software Design Engineer in Imagination Technologies Ltd., Pune.

While at the University of North Carolina at Charlotte, Chetan worked on many interesting software projects in Algorithms and Machine Learning as part of the academics. He received M.S. in Computer Science from the University of North Carolina at Charlotte in December 2017.

Currently, Chetan is a Software Engineer at the MathWorks Inc. in Natick, MA. He works on the popular technical computing softwares, MATLAB and Simulink, which are widely used in the academic, research field, and enterprises.