Multi-document Summarizer

Hazem Bakkar, Asma Al-Hamad and Mohammed Bakar

Abstract In this study, we address the multi-document summarization challenge. We proposed a summarizer application that implements three well-known multi-document summarization techniques; Topic-word summarizer, LexPageRank summarizer and Centroid summarizer. The contribution in this study is demonstrated by proposing a fourth summarization technique that is built on the previous acquired knowledge and experiments performed on the previously mentioned summarization techniques. Evaluating the system-generated summaries is performed using ROUGE [1], results showed that the new summarizer outperforms the other summarization techniques, and it takes a relatively short time to generate summaries comparing to other summarizers. However, LexPageRank summarizer evaluation performed better than the new summarizer evaluation, the cost of achieving a better evaluation using this technique was the time needed to generate the summaries, LexPageRank summarizer needs a long time to generate summaries comparing to other summarizers. In this study, DUC04 is used as a corpus in testing and implementing the proposed application.

1 Introduction

The massive size of textual data that exits online, and the huge number of documents that are available offline within various bodies of organizations raised the need to find effective techniques to extract the important information out of this enormous number of online and offline documents.

H. Bakkar (✉)
British University in Dubai, Dubai, UAE
e-mail: hazem.bakkar@hotmail.com

A. Al-Hamad · M. Bakar
Dammam University, Dammam, Saudi Arabia
e-mail: asma_toubassy@yahoo.com

M. Bakar
e-mail: mwb_sw@hotmail.com

© Springer International Publishing AG 2018

K. Shaalan et al. (eds.), *Intelligent Natural Language Processing: Trends and Applications*, Studies in Computational Intelligence 740,
https://doi.org/10.1007/978-3-319-67056-0_22

IR systems usually retrieve several number of documents that are related to the user query input, some other systems contain a document relevancy-assessment sub- system that retrieves several documents that are related to a user query. It is usual that the result of a query is hundreds of related documents. If we want to extract the informative text out of these documents, we will need an automatic summarizer. If we use a single document summarizer then it will most likely generate very similar summaries from the retrieved documents since most of them contain similar textual information. Here comes the benefit of using a multi-document summarizer, which will use the shared information that exist in the similar documents only once and then it will focus on the unique information that is spread in the cluster of the summarized documents.

2 Background

What is text summarization?

Text Summarization is generating a short text from a source longer text, while maintaining the main informative sentences of the source text or document. Text summarization methods can be classified into two types; abstractive summary and extractive summary.

2.1 Abstractive Summary

Abstractive summary is generating a summary depending on understanding the main concepts and information that exists in the source text. Abstract summarization uses Linguistics tools to examine and understand the text, these tools are used also to search for new expressions and concepts related to the information content in the source text, these expressions and concepts will be used to generate a short summary [2, 4]. In other words, Abstractive summary is about understanding the original text and generate a shorter version of it using other words.

The main problem in Abstractive summaries involves in the representation of the information that lies beneath the source text. The main concept of the abstractive summary is to understand the source text and then re-phrase it using new concepts and expressions. This task depends on the representation of the source text, which reflects the system understanding of the text content, the capability of generating a descent summary depends on the system's ability to capture the representation of the source text. It is known that until now, there are no systems that have an acceptable ability to understand natural language. Thus, abstractive summaries development is directly affected by improving the technology of intelligent under-standing of natural language [5].

2.2 Extractive Summary

This type of summaries depends on extracting key words or sentences from the original text using statistical methods to determine key text segments [6]. Extractive systems usually perform analysis for different surface level features such as term frequency, inverse document frequency, word location in the text, and indicators that point to the text segments that should be extracted [5, 6].

Extractive summary systems identify the most important text segments (word, sentence or paragraph) using two approaches: (1) by identifying the most frequent segment in the source text or (2) by being located in the most preferable position in the source text [6]. These approaches guarantee the simplicity of the extractive summary systems compared to the abstractive summary systems in both implementation and conceptual prospective [5].

Generating extractive summary includes two major stages [7]: (1) the preprocessing stage, and (2) the processing stage.

In the preprocessing stage, text is examined and analyzed by applying the following steps: firstly, the sentences boundary is identified. Different languages have different sentences boundaries, in English language, for example, the sentence boundary is identified by the dot at the end of the sentence. Secondly, removing stop-words that exist in the source text and removing any word with no semantics or words that do not represent relative information to the summary targeted content. Finally, stemming the words in the text; generating stems will insures the validation of the words semantics [7].

In the processing stage, features that determine the relevancy of sentences are chosen and computed, and then they are weighted using specific weight methods and equations. The sentences are given scores; the highest scored sentences will be added to the final summary [5].

Extractive text summaries have some drawbacks [8, 9]: (1) Extractive text summaries generally extract long sentences, which includes unnecessary or extra parts, this usually shrinks the available space allocated to more informative text segments which includes words or sentences. (2) Informative contents generally spread between sentences across the original text, this sparse of sentences makes it hard to capture the targeted content if the summary length is not long enough to capture and process these sentences. (3) Extractive summaries are weak in combining conflict information when generating the final summary. (4) Overall integrity in the final summary has some frequent problems, for example, sentences that contain pronouns; these pronouns usually lose their referents whenever they are extracted from their context. Another problem that faces the extractive summaries is in joining the sentences that are from different context, this will generate a faulty and inaccurate interpretation of the original text. Temporal expressions also lose their meaning when they are extracted from their context.

3 Multi-document Extractive Summarization

This approach uses a cluster of text documents that shares the same topic to generate a summary using extractive summarization techniques. The main goal of the multi-document summarization is to allow professionals and individuals to have an over-view about the topics and important information that exist in clusters of large number of documents within relatively a short time [5].

Single document summarization gives the reader a short compact summary about the contents of one summarized document. This type of summarization is limited by one source of information that is limited by one document. However, multi-document summarization generates a comprehensive summary from multiple sources; this type of summarization is considered more challengeable because of its variation of information sources. In 2003, Single document summarization track was removed from the Document Understanding Conference (DUC03), since that time the quantity of researches about single document summarization is remarkably shrinking [10]. Researches in the last decade proposed many multi-document summarization techniques, these techniques were heavily experimented and studied; in the following section, an implementation of a multi-document techniques will be discussed.

4 Implemented Summarization Techniques

In the proposed application three extractive summarization techniques were implemented and studied, in the next section, a theoretical review is presented to show the major parts of the summarization techniques that are crucial for the success of the proposed application.

4.1 Topic-Word Summary

Topic-Words Summarization is one approach of the Topic representation several approaches [11]. This type of summarization depends on identifying the most descriptive words in the input text, [12] used in his work a frequency threshold to determine the most descriptive words in the document in order to be selected for summarization. [12] ignored in his approach the most frequent words because it is most likely to be prepositions or determines. Also, he did not consider the words which appeared a few times in the document as these words represent the least important words in the text [11]. [13] proposed an enhanced statistical approach built on Luhun's approach, [13] used log-likelihood ratio test to determine the most descriptive words in the text, these words were called topic signatures as proposed in [14].

Topic signatures played a major rule in identifying the most important news content in multi document summarization as detailed in [15, 16]. The most important contribution in [13] approach that it determines a threshold that classifies the words in the input into two classes: descriptive and none descriptive. The classification process is accomplished by determining the statistical significance for the words in the input; the statistical significance will eliminate the need to use arbitrary thresholds in the proposed approach.

In order to determine the topic signature words, it is necessary to have a sufficient static information about the background corpus; this is mainly achieved by computing the frequency of mentioning the words in the background corpus. The likelihood of an input A and background corpus is calculated by assuming two cases: (1) The first case (H1) is when the probability of a word in the input text is equal to the probability of this word in the background corpus (B). (2) The second case (H2) is when a word in the input text has a different probability than the probability of the same word in the background corpus, usually this means that the probability of a word in the input text is higher than the probability of the same word in the background corpus [11]. The following is an arithmetic representation for the two cases [11]:

$$\mathrm{H1: P(w|A) = P(w|B) = p\ (w\ is\ not\ descriptive)}$$
$$\mathrm{H2: P(w|A) = pI\ and\ P(w|B) = pB\ and\ pI > pB\ (w\ is\ descriptive).}$$

To compute the likelihood of a text with reference to a descriptive word, the automatic summarizer uses binomial distribution. The input text and the background corpus are represented as a series of words w_i: w_1, w_2, w_n. The frequency of occurrences for each word is identified by Bernoulli trial with a success probability = p, the success is occurring only when $w_i = w$, the following distribution represents the over-all probability of observing the word w occurring k times in N trials [11].

$$b(k, N, p) = \binom{N}{k} p^k (1-p)^{N-k} \tag{1}$$

In H1, p is computed from the input text and the background corpus together, while in H2, p1 is computed from the input text only, and p2 is computed from the back-ground corpus. Now all the elements are ready to compute the likelihood ratio using the following equation:

$$\lambda = \frac{b(k, N, p)}{b(k_I, N_I, p_I) \cdot b(k_B, N_B, p_B)} \tag{2}$$

where the terms with subscript I is calculated from the input text, and the terms with index B are calculated from the background corpus [11].

Using Eq. 2, we can represent $-2\log \lambda$ by the statistical distribution X^2, this will make it possible to identify the words that are considered as topic signatures [11]. Regarding the term topic signatures, we can determine the importance of a sentence in the summarizer input by calculating the number of topic signatures that exists in the sentence or by identifying the part of the sentence that is occupied by topic signatures [11]. These two ways of identifying the importance of a sentence are forming the scoring functions that are used to weight the sentences in the summarization process. It is noted that [12] approach gives a higher score for long sentences, while [13] approach gives higher score to sentences that have a higher density of topic signatures [11].

4.2 Centroid Summary

A centroid is a collection of words that are considered descriptive and important for a cluster of documents. Centroids main benefit is that they can determine whether a document in a cluster is relevant or irrelevant, also Centroids can identify the salient sentences in a cluster of documents [17].

Normally, documents main subjects are describing different topics in a sequential manner, a topic after another. This arrangement of topics is also implemented when generating a summary for any collection of documents. In order to build a meaningful summary; documents are grouped in clusters depending on their topic, so that every cluster of documents is addressing a relevant topic [5]. Documents in these clusters are represented as Term-Frequency-Inverse Document frequency (TF-IDF) vectors [18]. Term Frequency (TF) is representing the average of times a term occurs in each document in the cluster while Inverse Document Frequency (IDF) is calculated using the documents in all the clusters that make the corpus. Each cluster of documents in the corpus is considered a separate theme, and these clusters are representing the required input for the automatic summarizer that is under study. The theme that is derived from the cluster consists the words that have the highest TF-IDF scores in that specified cluster.

In Centroid summary, there are three main factors that determine whether sentences will be selected to be added to the final summary or not [5]; the first factor is computing the similarity of the sentences with the theme of the cluster (C_{i}). The second factor is the location of the sentence in the original input text (L_{i}). Different contexts of documents have different weights for sentence location importance in the body of the text, for example, in the news context, the sentences that are closer to the beginning of the text gain higher scores as they have higher importance. These sentences will likely have more chances to be added to the final summary. The third factor that determine the score that is assigned to a sentence is the similarity between this sentence and the first sentence of the document that contains both sentences (F_{i}) [5]. The following equation calculates the overall score (S_i) of the sentence (i) [5]:

$$S_i = W_1 * C_i + W_2 * F_i + W_3 * L_i \tag{3}$$

Ci, Li and *Fi* are the scores that are described in the previous paragraph, while W_1, W_2 and W_3 are the weights for the linear combination of the *Ci, Li, Fi* scores.

In centroid summary each document in the cluster is identified as a weighted TF-IDF vector, then a centroid is created using the first document in the cluster, after that the TF-IDF for each other documents that will be processed by the summarizer will be compared with the centroid generated from the first document of the cluster. If the similarity measure between a document and the centroid is within a specified threshold then the document under process will be added to the cluster for further processing, otherwise the document will be ignored. The following equation is used to calculate the similarity between the centroid and the processed document [19]:

$$sim(D, C) = \frac{\sum_k (d_k * c_k * idf(k))}{\sqrt{\sum_k (d_k)^2}\sqrt{\sum_k (c_k)^2}} \tag{4}$$

4.3 LexPageRank Summary

In this type of summaries, the selection of sentences that build up the summary is based on selecting the most central sentences in the cluster of documents that contains the most informative words. This approach of selecting the sentences is the same in principle of the selection procedure in centroid summary. The difference between LexPageRank summary and centroid summary is in the way of measuring the sentence centrality. LexPageRank summary is using Prestige principle to determine the most central sentences in a cluster of documents.

In LexPageRank summary, the sentences of a document are represented as a graph of TF-IDF vectors [2, 3] and a cluster of documents is represented as a network graph of sentences. Some of these sentences are similar to each other, and some other sentences have a small degree of similarity between them, which means that these sentences share a little amount of information between each other [2, 3]. Prestigious sentences are the sentences that are similar to many other sentences in the cluster; these sentences are considered as the most central sentences with reference to the topic [2, 3]. In order to identify the most prestigious sentences, similarity metric is used to define centrality degree for a sentence; formally, cosine similarity is used to achieve this goal. A cluster of documents can be represented by a cosine similarity matrix, where each element in the matrix represents a cosine similarity value between a sentence and its corresponding sentence. There are two methods to compute sentence prestige using the cosine similarity matrix: 1—Degree centrality. 2—Eigenvector centrality and LexPageRank [2, 3].

Degree Centrality.
It is most likely to observe relevancy between sentences in the related documents in a certain cluster, this means that cosine similarity values will be mostly more than zero in the cosine similarity matrix. A threshold value is determined in order to ignore the low values of cosine similarity in the matrix; this threshold will guarantee considering the significant cosine similarity values for sentences, these sentences contain the information that will most likely be considered for adding in the final summary. After deleting the low values of cosine similarity, the cluster can be implemented as an undirected graph where the sentences are represented as nodes, and these nodes have connections between each other, these connections represent the significant similarity between the connected sentences. Now degree centrality of a sentence can be identified as the degree of each node in the similarity graph [2, 3]. The value of cosine similarity threshold has a direct effect on the generated summary. If this threshold is too small then weak and less informative sentences can appear in the produced summary, while choosing a high value of cosine similarity will cause losing informative and important sentences that should be added to the summary.

Eigenvector Centrality and LexPageRank.
In computing the degree centrality in a cluster of documents, each edge is considered as a vote, a node is considered a prestigious one when it has high number of votes. The voting approach has one important drawback, this drawback appears when there is a cluster of related documents; and this cluster contains one document which is not related to the main topic of the cluster. When using voting approach within the document, some of the sentences in this document will get high voting degree and will be considered prestigious, this voting degree will qualify these sentences to be added to the final summary. This means that the produced summary will contain some unrelated sentences, which should not be considered in this summary. This drawback can be avoided by considering the prestigious degree of the node that did the voting, and considering the node prestige in weighting other nodes [2, 3].

PageRank is an important well-known implementation of the prestige principle [20]. PageRank is the tool for assigning a prestige score for each webpage in the web, the score in the PageRank is calculated by counting the number of pages that link to that page and the individual scores of the linking pages [2, 3]. The PageRank of page A is calculated as follows:

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)}) \tag{5}$$

whereT_1 … T_n are pages that link to page A, $C(T_i)$ is the number of the outbound links from page T_i, d is the damping factor which ranges from 0 to 1 [2, 3]. To calculate the value of PageRank, a binary adjunct matrix M is constructed, where M (u, v) = 1 when there is a link from page u to page v, then the matrix is normalized so that the summation of each row is equal 1, after that the principal eigenvector of the matrix is calculated. Thus, PageRank of *i*-th page equal to the *i*-th element in the eigenvector [2, 3].

The previous procedure can be implemented on the cosine similarity matrix to identify the most prestigious sentences in the document. PageRank is used to estimate the weight for each vote by considering the prestige degree of a sentence, the more a sentence is prestigious the higher weight its vote will gain. Since cosine similarity is symmetric then the graph that represent the sentences is undirected graph, this will have no influence on the way of calculating the principal eigenvector [2, 3]. Principal Eigenvector is computed using a simple iterative power method [2, 3].

5 The Proposed Summarizer Application

In Sect. 4, three well-known automatic summarization techniques—Centroid summary, Topic-word summary and LexPageRank summary—were reviewed. In our project, these three techniques will be implemented in addition to a fourth summarization technique that is designed by the authors depending on the knowledge acquired from studying several summarization techniques, the following section will describe the implementation of these techniques.

5.1 Centroid Summarizer

Centroid summarizer is based on identifying the most similar sentences to the original document, for this purpose, the following equation is used in our project:

$$Centrality = \frac{1}{N}\sum\nolimits_{y \neq x} sim(x, y) \tag{6}$$

where x and y are vectors that represent each sentence in the input. Words in the input represents features, the value of the feature is equal to the weight of this word in the vector. Weights are calculated using Term Frequency (TF) or (TF-IDF) or by using binary representation in which the value will be 1 if the term appears in the sentences or 0 if the term does not exist in the sentences. To correctly implement any summary technique in the proposed project, it is necessary to determine the following parameters and methods: (1) Vector feature representation. (2) Similarity approach. (3) Sentence length limitation. (4) Redundancy mitigating method.

In implementing the centroid summarizer, Binary representation is used to represent vector feature weights, because it is simple to compute and it showed strong results in the testing phase. In this summarizer, cosine similarity metric is used to identify the similarity between the vectors. The sentence length in this summarizer is ranged between 15 and 50 words with a total summary length no more than 200 words maximum; these words were tokenized using NLTK.

To decrease the redundancy of sentences, any sentence has a cosine similarity more than 0.75 with a sentence in the summary is rejected. The values of these parameters were chosen carefully after performing several trials on the proposed application.

After calculating the centrality score, the program will select the sentences that will form the centroid summary, for this purpose the program will use a greedy algorithm that will choose the sentences that have the highest score then the second highest, and so on until the formed summary reaches the words count number limitation. The following pseudo code describes the greedy algorithm that is used to generate the final summary. This algorithm is also used in the other summarization techniques that are implemented in this application [21].

```
Greedy (Sentences, threshold), Begin
Centrality = [Sim (Sen, Doc) for Sen in Sentences]
Build the Sentence + Centrality Dictionary Diction
(we call it Diction)
Sort Diction according to Centrality in decreasing order.
Current_Summary = []; length = 0;
while((len < threshold)and(i <= Diction.size())

{if (Valid(Diction[i], Current_Summary))
{Current_Summary.append(Diction[i].Sentence)
length += len(Diction[i].Sentence)
}

i += 1
}
print(CurrentSummary) End
```

5.2 Topic-Word Summarizer

In this summarizer, the importance of sentences is computed by counting the topic words in each sentence, this will determine the weights for words. To score the sentences with respect to topic signatures, several equations can be used:

$$TWeights\ (S) = \#\ of\ topic\ words\ in\ sentence\ x \tag{7}$$

$$TWeights\ (S) = \frac{\#\ of\ topic\ words\ in\ sentence}{\#\ of\ words\ in\ sentence\ x} \tag{8}$$

In this project, the following equation is used to represent the sentence vector Weights:

$$TWeights\ (S) = \frac{\#\ of\ topic\ words\ in\ sentence\ x}{\#\ of\ nonstopwords\ in\ sentence\ x} \tag{9}$$

The reason for selecting Eq. 9 as a representation method in this summarizer is that it avoids the negative effect of counting stop words in calculating the weights. Also, this equation will maximize the productive use of the allowed number of words in the generated summary.

A topic tool algorithm is used to determine the topic words in the input text, this tool is developed by [21], and it is implemented in the *topics.py* file. This tool uses a cut off parameter for Topic words, the default value for this parameter is 0.1, this value was tested and showed better results than other values like 0.2 and 0.3.

The generated summary will include no more than 200 words with sentences' lengths between 15 and 50 words maximum, any sentence with cosine similarity more than 0.75 will be ignored.

5.3 LexPageRank Summarizer

Every sentence in this type of summarizers is represented as a node. The similarity between two nodes is represented by an edge if this similarity is exceeding a predetermined threshold, otherwise, there will be no edge between these two nodes.

In the proposed project the LexPageRank summarizer uses TF-IDF to represent the vectors, this generates more accurate vectors and relatively better results in the evaluation phase using ROUGE [1]. The similarity threshold used in this summarizer is equal to 0.2, this value showed good results when used in the experiment performed in [2, 3], another reason for using this threshold is that building LexPageRank summary is a time-consuming process, it is too slow, so that it was difficult to try several similarity thresholds.

The ending condition in the LexPageRank summarizer is that the iteration will end only if all values become less than 0.001 between iterations. This value showed that it is a reasonable value that creates a balance between performance and good results. It is noted that the quality of results did not noticeably changed when decreasing the threshold.

The sentence length in this summarizer is ranged between 15 and 50 words; the words were tokenized using NLTK. To decrease the redundancy of sentences, any sentence has a cosine similarity more than 0.75 with a sentence in the summary was rejected, and the generated summary is limited to maximum 200 words.

5.4 The Proposed Summarizer

After studying several techniques about automatic multi-documents summarizers, an idea about an additional summarization technique is formed, the proposed summarization technique uses the first and the last sentences of each input document as a reference to build the final summary. The proposed summarizer weights and scores sentences using the same technique in Centroid Summarizer, also the proposed summarizer is using topic-words approach to determine which words are considered topic signatures, it uses the same technique in Topic-words summarizer. The binary representation is the method that is used for representing sentences in this summarizer, and Eq. (9) is used to weight the sentences exactly the same way performed in the topic-words summary.

In this summarizer, the similarity measurement and the scoring processes are performed with regard to the entire cluster not only the selected subset of sentences that was collected at the beginning of the summarization operation.

After representing the sentences and scoring them, the sentences validity is verified by counting the number of words that exists in each sentence, which should be between 15 and 50 words, any sentence has a number of words that is not within this limitation will be ignored, also, any sentence with Cosine similarity value more than 0.75 will be ignored. Since the main idea behind this project is to implement extractive summarizers, it was a good idea to add some aspects of the other type of summarization methods, which is the abstractive summarization technique. This idea was achieved by replacing the least frequent nouns and verbs with their synonyms, by doing this; the final summary will have an extra benefit especially in readability and quality. It is meant to keep the frequent verbs and nouns unchanged because these words have a strong effect on the final summary, changing these words maybe changes the meaning or context of the sentences which can mislead the summarizer to generate an inaccurate summary with wrong meanings and context.

The following steps describe the proposed summarizer's operation:

1. Remove the stop words from the input text.
2. Tokenize sentences, and then extract the first and last sentence of the input document.
3. Load topic-words from *.ts, these files are generated using topic identifying tool called TopicS which is implemented in the file Topics.py [21].
4. Represent the sentences as vectors.
5. Get the Cosine similarity between the extracted vectors and the rest of the sentences in the entire cluster.
6. Scoring the sentences using the Cosine similarity metric.
7. Weighting and scoring the sentences using Eq. 9.
8. Combine both metrics results in one value that represents the overall score.
9. Sort the scored sentences descending.

10. Greedily select the sentences with the highest score then the second highest and so on until the generated summary reaches the words allowed limit which is 200 words. Also, all sentences with Cosine similarity more than 0.75 will be ignored.
11. Replace the least frequent nouns and verbs with their synonyms.
12. Generate the final summary.

6 Running the Application

To run the application, the user has to choose which type of summarizer he wants to use; this can be done by activating the desired instruction by deleting the hash sign (#) from the code line in the file project.py.

The following code lines exist at the end of the file project.py:

```
# summary = centrality_sum(path)
#summary = topic_word_sum(path, topicFile)
# summary = lex_rank_sum(path)
# summary = custom_summarizer(path, topicFile)
```

The system-generated summaries will be saved automatically in a folder called summaries; this folder exists in the parent folder that contains the file project.py.

7 Software, Dataset, and Tools Used in the Project

In the following sections, the main software and tools used in the project are generally reviewed, in addition to a description of the corpus used in implementing the summarizer application.

7.1 Python 2.7.6

Python is one of the most flexible programming languages that is known with the following aspects and specifications:

1. Python is an open source language that is free for everyone; also, it is an object-oriented language.
2. Python is an interpreted language that can be easily understood, it is also a dynamic language that can represent and manipulate massive types of variables and elements.

Python is usually competing with other languages like PERL, Java or TCL.

Python is very flexible language, because it has dynamic data types that are accompanied by dynamic typing scheme. In addition, Python can use Modules that are created by other languages like C++ [22].

Python most valuable aspects are: (1) it is readable. (2) It is clear and a powerful language. (3) It is used in a lot of different domains like web applications, system administration, desktop applications, windows applications and scientific research [23].

7.2 Natural Language Toolkit (NLTK)

NLTK is a premier platform for developing applications and systems that process human natural language data. NLTK contains more than 50 corpora and lexical resources. NLTK contains a large number of text-processing libraries for semantic reasoning, tokenization, and stemming. It also has many other libraries and functions that manipulate natural language data [24].

NLTK is a free text-processing platform that can be used by different operating systems like Windows, Mac, and Linux [24].

7.3 Dataset Used in the Project

DUC04 stands for Document Understanding Conference. This is one of the most famous conferences for researches about summarization related topics, it is held annually by the National Institute of Standards and Technology (NIST) [25]. In the proposed project DUC04 corpus is used to test the application functionality; it is also used to evaluate the quality of the generated summary by the four summarization techniques that are implemented in the project [26].

DUC04 corpus contains 50 clusters of news text documents; the documents in each cluster mainly discuss the same topic. Every cluster contains an average of 10-text documents. DUC04 also contains Model summaries that are made by human summarizers; these models are used for evaluating the quality of the system-generated summaries.

7.4 Rouge

In this research, ROUGE is used to perform the evaluation for the summaries generated using the proposed summarizer [1].

ROUGE is the abbreviation for (Recall Oriented Understudy for Gist Evaluation), it is the official scoring technique for Document Understanding Conference (DUC) 2004 [1].

ROUGE is used for English Multi-document Summarization. In ROUGE the accuracy of the system-generated summaries are measured by comparing the overlap between the Model summaries that are written by professional human summarizers and the system-generated summaries [1].

ROUGE uses different measures, ROUGE-N uses N-Grams to measure the overlap, ROUGE-L uses Longest Common Subsequence, and ROUGE-W uses Weighted Longest Common Subsequence [1].

8 Summarizer Performance

The quality of the automatic generated summary is identified by comparing this summary with the human-made summaries that share the same input documents.

ROUGE is used to evaluate the performance of the application. ROUGE uses several parameters to evaluate the summarizer performance. The same parameters will be used for all of the summarizing techniques implemented in this project. Previous researches used ROUGE for evaluating summarizers showed that the best settings for ROUGE in evaluating summaries are ROUGE-2 Average-R score (Recall), with words stemming and without removing the stop words [27]. This setting is implemented using the following instruction in ROUGE:

```
/ROUGE-1.5.5.pl –c 95 –r 1000 –n 2 –m –a -1 100 –x config.xml
```

The evaluation of the application performance was executed using two stages, the first stage was testing the summarizers using one folder of DUC04 corpus, the benefit of this stage is to get an overview about how well the system summarizers are performing. Evaluating one folder gives us quick results, while performing the evaluation on the entire corpus will take at least 6 hours, which is a non-practical process in our case; because an evaluation using ROUGE should be done everytime after changing the parameters in the summarizers in the sake of improving the quality of the system-generated summaries.

The results of the tests that were performed using ROUGE on a single folder from DUC04 corpus are shown in Table 1, Higher scores means a higher similarity to model summaries which are prepared by human judges, therefore, higher scores mean a better summary quality.

The results of the tests that were performed using ROUGE on the entire corpus (DUC04) are shown in Table 2:

Table 1 ROUGE results based on testing one folder from the corpus

Technique	ROUGE-2 recall	ROUGE-1 recall
Centroid tech	0.11414	0.44226
Topic-word tech	0.10670	0.43735
LexPageRank	0.11911	0.43243
New technique	0.04963	0.32896
Baseline summaries	0.04963	0.41278

Table 2 ROUGE results based on testing the entire corpus

Technique	ROUGE-2 recall	ROUGE-1 recall
Centroid tech	0.04409	0.30884
Topic-word tech	0.03547	0.2916
LexPageRank	0.6987	0.34126
New technique	0.05882	0.32896

The results of the trials showed that testing the entire corpus gave better results than the trials performed on a single directory. The proposed summarizer outperforms all other techniques presented in this project except for LexPageRank summarizer which showed better results than the new summarization technique. However, testing using LexPageRank summarizer took more than 4 hours to test only 14 folders out of 50 folders, the same applies for generating summaries using Centroid summarization technique. Therefore, it is clear that there is a penalty for producing high quality summaries; this penalty is consuming a long time to generate the summary.

9 Conclusion

In this project, we implemented three famous summarization techniques Topic-word summarizer, LexPageRank Summarizer and Centroid summarizer. Then a new fourth summarization technique is proposed and implemented. The evaluation of the system-generated summaries using the four summarization techniques-shown in Tables 1 and 2—clearly indicated an excellent ROUGE score when compared to the baseline summaries which are made by human summarizers. Also, it is noted that evaluating the summarizers' performance using a single folder instead of the entire corpus showed a less quality than performing the same evaluation using the entire corpus. This means that the overlap between the system-generated summaries and the model summaries is increased after we extend the size of the summarized documents to include the entire corpus; this is because of the use of the unique information that is spread between the documents in the entire corpus.

ROUGE evaluation results also showed that LexPageRank summarizer achieved the best score between the other three summarizers, however LexPageRank summarizer also showed that it needs a long time to generate the summaries from the corpus that is used in the project, it took more than 6 hours to generate the summaries. In the other hand, the new proposed summarizer and the topic-word summarizer took less than 10 min to generate the summaries out of the entire (DUC04) corpus. Centroid summarizer also needs a long time-relatively-than Topic-word summarizer and the new proposed summarizer to build its summaries.

10 Future Work

The summarizer project has many aspects that can be modified and extended. The proposed project summarizes only English language text documents, as a future work the proposed application can be modified to summarize Arabic text documents as a first step, then it can be extended to summarize other languages texts. Another addition that is considered crucial for the summarizer project is to develop a graphical user interface; this will make the use of the application easier for end users. If the proposed application is equipped with a graphical user interface, then it will be easy to put it online and make it available for public to use and test.

Evaluating the system-generated summaries is very important to determine the quality of these summaries, so it will be helpful if the application can make an automatic summary evaluation, so that the user can estimate which technique can give better results for summarizing his cluster of documents.

References

1. Lin, C.: Rouge: a package for automatic evaluation of summaries, pp. 74–81 (2004)
2. Erkan, G., Radev, D.: LexPageRank: prestige in multi-document. Text Summ. **4**, 365–371 (2004)
3. Erkan, G., Radev, D.: LexRank: graph-based lexical centrality as salience in text summarization. J. Artif. Intell. Res. (JAIR) **22**(1), 457–479 (2004)
4. Hahn, U., Romacker, M.: The SYNDIKATE text knowledge base generator, pp. 1–6 (2001)
5. Gupta, V., Lehal, G.: A survey of text summarization extractive techniques. J. Em. Technol. Web Intel. **2**(3), 258–268 (2001)
6. Kyoomarsi, F., Khosravi, H., Eslami, E., Dehkordy, P., Tajoddin, A.: Optimizing text summarization based on fuzzy logic, pp. 347–352 (2008)
7. Gupta, V., Lehal, G.: A survey of text mining techniques and applications. J. Em. Technol. Web Intel. **1**(1), 60–76 (2009)
8. Lin, J.: Summarization. In: Encyclopedia of Database Systems, 1st edn. Springer, Heidelberg, Germany (2009)
9. Cheung, J.: Comparing abstractive and extractive summarization of evaluative text: controversiality and content selection. B. Sc. (Hons.). University of British Columbia (2008)
10. Alliheedi, M.: Multi-document Summarization System Using Rhetorical Information. Master of Mathematics. The University of Waterloo
11. Nenkova, A., McKeown, K.: A survey of text summarization techniques. Springer, pp. 43–76 (2012)
12. Luhun, H.: The automatic creation of literature abstracts. IBM J. Res. Dev. **2**(2), 159–165 (1958)
13. Dunning, T.: Accurate methods for the statistics of surprise and coincidence. Comput. Linguist. **19**(1), 61–74 (1993)
14. Lin, C., Hovy, E.: The automated acquisition of topic signatures for text summarization, pp. 495–501 (2000)
15. Conroy, J., Schlesinger, J., O'Leary, D.: Topic-focused multi-document summarization using an approximate oracle score. In: the COLING/ACL on Main conference poster sessions. Association for Computational Linguistics, pp. 152–159 (2006)

16. Harabagiu, S., Lacatusu, F.: Topic themes for multi-document summarization, pp. 202–209 (2005)
17. Radev, D., Jing, H., Styś, M., Tam, D.: Centroid-based summarization of multiple documents. Inf. Process. Manag. **40**(6), 919—938 (2004)
18. Zhang, Y., Zincir-Heywood, N., Milios, E.: Narrative text classification for automatic key phrase extraction in web document corpora, pp. 51–58 (2005)
19. Radev, D., Hatzivassiloglou, V., McKeown, K.: A description of the CIDR system as used for TDT-2, p. 205 (1999)
20. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: bringing order to the web. Stanford InfoLab (1999)
21. Nenkova, A.: Computational Linguistics (2012)
22. Haas, J.: Python. http://linux.about.com/cs/linux101/g/python.htm
23. Voidspace.org.uk: A Very Brief Introduction to Python. http://www.voidspace.org.uk/python/articles/python_datatypes.shtml#introducti
24. Nltk.org: Natural Language Toolkit—NLTK 3.0 documentation. http://www.nltk.org/
25. www-nlpir.nist.gov.: Document Understanding Conferences—Introduction. http://www-nlpir.nist.gov/projects/duc/intro.html
26. Hong, K., Conroy, J., Favre, B., Kulesza, A., Lin, H., Nenkova, A.: A repository of state of the art and competitive baseline summaries for generic news summarization. In: Proceedings of LREC, May (2014)
27. Owczarzak, K., Conroy, J., Dang, H., Nenkova, A.: An assessment of the accuracy of automatic evaluation in summarization, pp. 1–9 (2012)