A NEWLY PROPOSED TECHNIQUE FOR SUMMARIZING THE ABSTRACTIVE NEWSPAPERS' ARTICLES BASED ON DEEP LEARNING

Sherif Kamel Hussein[1] and Joza Nejer AL-Otaibi[2]

[1]Associate Professor – Department of Communications and Computer Engineering
October University for Modern Sciences and Arts, Giza- Egypt
[1,2]Head of Computer Science Department, Arab East Colleges for Graduate Studies, Riyadh, KSA

ABSTRACT

In this new era, where tremendous information is available on the internet, it is of most important to provide the improved mechanism to extract the information quickly and most efficiently. It is very difficult for human beings to manually extract the summary of large documents of text. Therefore, there is a problem of searching for relevant documents from the number of documents available, and absorbing relevant information from it. In order to solve the above two problems, the automatic text summarization is very much necessary. Text summarization is the process of identifying the most important meaningful information in a document or set of related documents and compressing them into a shorter version preserving its overall meanings. More specific, Abstractive Text Summarization (ATS), is the task of constructing summary sentences by merging facts from different source sentences and condensing them into a shorter representation while preserving information content and overall meaning. This Paper introduces a newly proposed technique for Summarizing the abstractive newspapers' articles based on deep learning.

KEYWORDS

Abstractive Text Summarization, Natural Language Processing, Extractive Summarization, Automatic Text Summarization, Machine Learning.

1. INTRODUCTION

Nowadays with the dramatic growth of the Internet, there is an explosion in the amount of text data from a variety of sources. That may lead to an expansion for the availability of the documents that need an exhaustive research in the area of automatic text summarization [1]. The volume of text is considered as an invaluable source of information and knowledge, which needs an effective summarization to be useful. Text summarization is the problem of creating a short, accurate, and fluent summary of a longer text document [1].Automatic text summarization methods are greatly needed to address the ever-growing amount of text data that are available online to discover and consume the relevant information faster [2]. Text Summarization methods can be classified into extractive and abstractive summarization [3]. An extractive summarization method consists of selecting important sentences, paragraphs etc. from the original document and concatenating them into shorter form. An Abstractive summarization is based on understanding the main concepts in a document and then express those concepts in clear natural language [3].

There are many reasons why Automatic Text Summarization is useful [4]:

- Summaries reduces reading time.
- When researching documents, summaries make the selection process easier.
- Automatic summarization improves the effectiveness of indexing.
- Automatic summarization algorithms are less biased than human summarizers.
- Personalized summaries are useful in question-answering systems as they provide personalized information.
- Using automatic or semi-automatic summarization systems enables commercial abstract services to increase the number of text documents that are able to process.

Automatic text summarization is a common problem in machine learning and natural language processing (NLP). Skyhoshi, who is a U.S.-based machine learning expert with 13 years of experience and currently teaches people his skills, said that "the technique has proved to be critical in quickly and accurately summarizing voluminous texts, something which could be expensive and time consuming if done without machines." Machine learning models are usually trained to understand documents and distill the useful information before extracting the required summarized texts. With such a big amount of data circulating in the digital space, there is a need to develop machine learning algorithms that can automatically shorten longer texts and deliver accurate summaries that can fluently pass the intended messages. Applying text summarization reduces reading time, accelerates the process of researching for information, and increases the amount of information that can fit in an area [5]. In this study,the machine-learning approach will be used to conduct a model to produce an abstractive text summarization based on Python programming language.

1.1. Main Steps for Text Summarization

There are three main steps for summarizing documents. These are topic identification, interpretation and summary generation [3]. First step, is Topic Identification which identifies the most prominent information in the text. There are different techniques for topic identification as Position, Cue Phrases and word frequency. Methods which are based on that position of phrases are the most useful methods for topic identification. Second Step, is the Interpretation. In This step, different subjects are fused in order to form a general content. Finally the third step is the Summary Generation in which, the system uses the text generation methods.

2. LITERATURE REVIEW

2.1. Abstractive Summarization with the Aid of Extractive [6]

This technique proposed a general framework for abstractive summarization which incorporates extractive summarization as an auxiliary task by composeding of a shared hierarchical document encoder, an attention-based decoder for abstractive summarization, and an extractor for sentence-level extractive summarization. Learning these two tasks jointly with the shared encoder allow to better capture the semantics in the document. Furthermore, experiments on the Cable News Network (CNN)/Daily Mail dataset demonstrate that both the auxiliary task and the attention constraint contribute to improve the performance significantly, and their model is comparable to

the state-of-the-art abstractive models. In particular, general framework of their proposed model is composed of 5 components namely: word-level encoder encodes the sentences word-by-word independently, sentence- level encoder encodes the document sentence-by-sentence, Sentence extractor makes binary classification for each sentence, Hierarchical attention calculates the word-level and sentence-level context vectors for decoding steps, Decoder decodes the output sequential word sequence with a beam-search algorithm.

2.2. Toward Abstractive Summarization using Semantic Representations Arxivpreprint, Arxiv: [7]

This technique presented a novel abstractive summarization framework that draws on the recent development of a treebank for the Abstract Meaning Representation(AMR).In this framework, the source text is parsed to a set of AMR graphs, the graphs are transformed into a summary graph, and then text is generated from the summary graph. They focus on the graph-to graph transformation that reduces the source semantic graph into a summary graph, making use of an existing AMR parser and assuming the eventual availability of an AMR-to text generator. The framework is data-driven, trainable, and not specifically designed for a particular domain. Experiments based on gold standard AMR annotations and system parses show promising results.

2.3. Abstractive Document Summarization Via Bidirectional Decoder [8]

This technique introduced an abstractive document summarization via bidirectional decoder. It is based on Sequence-to-sequence architecture with attention mechanism which is widely used in abstractive text summarization, and has achieved a series of remarkable results. However, this method may suffer from error accumulation. It proposed a Summarization model using a Bidirectional decoder (BiSum), in which the backward decoder provides a reference for the forward decoder. The authors used attention mechanism at both encoder and backward decoder sides to ensure that the summary generated by backward decoder can be understood. Experimental results show that the work can produce higher-quality summary on Chinese datasets Transport Topics News (TTNews) and English datasets Cable News Network (CNN)/Daily Mail.

2.4. A Comprehensive Survey on Extractive and Abstractive Techniques for Text Summarization [9]

In this technique the generation of summary is done by understanding the whole content and representing it in its own terms. This is achieved using a Recurrent Neural Network consisting of Gated Recurrent Unit (GRU) or Long Short-Term Memory (LSTM) cells. The author highlights the recent abstractive techniques used for text summarization and provides information on the standardized datasets in addition to testing methodologies that are used to evaluate the performance of the system. For instance, structured approach fundamentally encodes the most indispensable information from the document(s) through mental blueprints like layouts, extraction principles, and elective structures like tree, ontology, rule, and graph structure. On the other hand, in tree-based approach, sentences from multiple documents are clustered according to the themes they represent. Second, these themes are re-ranked, selected, and ordered according to their significance. The formed syntactic trees are subsequently merged using Fusion Lattice Computation to assimilate information from different themes. Linearization is carried out for the formation of sentences from the merged tree using Tree traversal.On the other hand, Graph-Based Approach uses the graph data structure for language representation. The sentence formation is subjected to constraints such as, it is mandatory to have a subject, verb, and predicate in it. Along with this, a compendium is used for Linguistic and Summary Generation purposes. General

notion in Extractive Text Summarization is to weight the sentences of a document as a function of high-frequency words, disregarding the very high frequency common words. Location Method exploits the idea of identifying important information in certain part of context. Sentence extraction should be possible utilizing Neural Network Architectures. One of these strategies is a classifier which includes the navigation of the archive consecutively, and choosing whether to include the sentence into the rundown. Extractive techniques include picking sentences in an arbitrary way. Extractive text summarization is conducted using neural model. The advantage of using this method over the traditional pure mathematical and Natural Language Processing (NLP) techniques is to understand more contexts. The models improve the depiction of sentences by combining the important sentences to shorten the size and maintain the semantics at the same time.

2.5. Neural Extractive Summarization with Side Information, Arxiv Preprint, Arxiv: [10]

This Technique was implemented by (Narayan et al.) to achieve summarization using sentence extraction. The model was based on an encoder—decoder architecture consisting of Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). The idea behind it was to acquire portrayals of sentences by applying single-layer convolutional neural systems over sequences of words embedding and using the recurrent neural network to make groupings out of sentences. CNNs were used to capture the important patterns amongst the sentences in the article. This CNN was used to extract the sequence of words in a sentence. The document encoder was used to identify the sequence of sentences to get the document representation. Abstractive Summarization has been achieved using a sequence to sequence encoder–decoder model. This model has its famous application in Neural Machine Translation. It is essentially used to preserve the dependencies LSTM cells that are used. These cells are the most atomic unit of an encoder and decoder and better method to address the problem of dependency. Time Delay Neural Network (TDNN) is used to achieve this along with max pooling layers. TDNN receives the output of the bottom layer to the input layer. This helps in preserving the dependency over long sentences.

2.6. Abstractive Multi-Document Text Summarization using a Genetic Algorithm [11]

This Technique Introduced an abstractive multi-document text summarization using a genetic algorithm (MDTS) .It is based on Multi-Document Text Summarization (MDTS), which consists of generating an abstract from a group of two or more number of documents that represent only the most important information of all documents. In this article, the authors proposed a new MDTS method based on a Genetic Algorithm (GA). The fitness function is calculated considering two text features: sentence position and coverage. They proposed the binary coding representation, selection, crossover and mutation operators to improve the state-of-the-art results. The proposed method consists of three steps. In the first step, the documents of the collection were chronologically ordered, then the original text is adapting to the entry of the format of the GA, where the original text is separated in sentences. Also, the text pre-processing is applied to the collection of documents. Firstly, the text was divided into words separated by commas, then some tags were placed in the text to be able to differentiate quantities, emails, among others.

3. THE NEWLY PROPOSED SYSTEM

3.1. Overview

The newly proposed System for summarization based on machine learning technique is constructed based on extracting key phrases from sentences and then using a deep learning model to learn the collocation of phrases.

3.2. Block Diagram of the Newly Proposed System

The process flow of the proposed system is divided into three steps: text pre-processing, phrase extraction and text summary generation as shown in Fig 1.

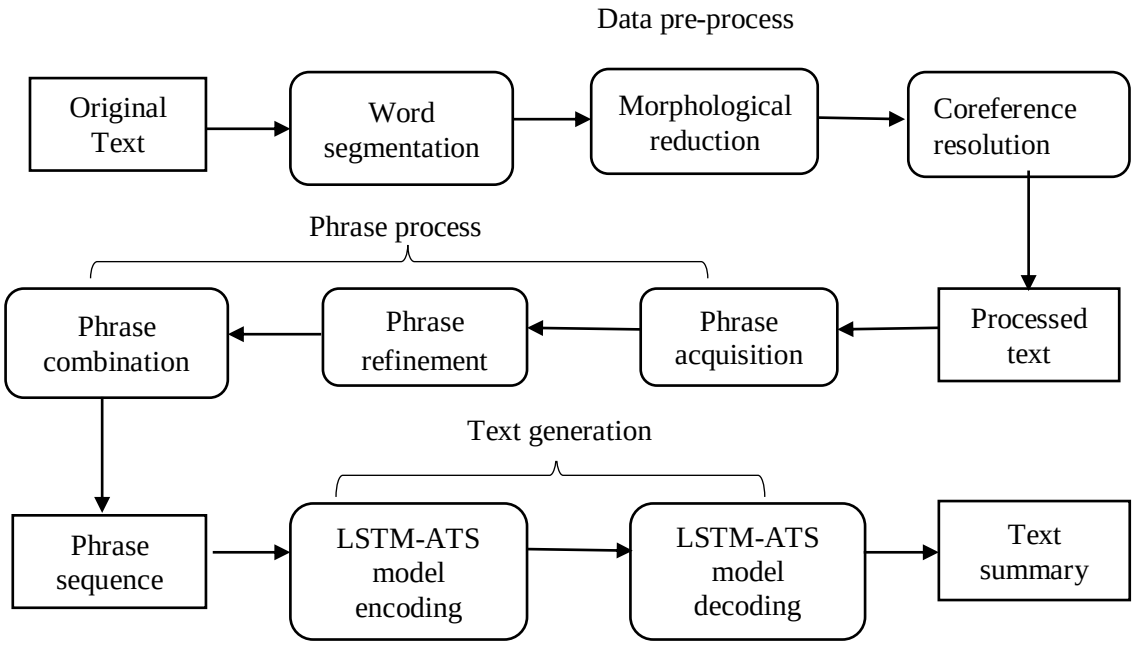

Figure1. System block diagram

Data pre-processing phase includes, word segmentation, morphological reduction, and coreference resolution. After word segmentation, the morphological reduction will be applied to merge these phrases into one. In the original text there are pronouns and demonstratives that will cause ambiguity during model training, so the coreference resolution will be used before the model is trained. Phrase extraction includes three sub-steps: phrase acquisition, phrase refinement, and phrase combination. Phrase extraction method is used to acquire phrases. Phrase refinement is applied to remove the redundancy in semantics or syntactic structure in the extracted phrase before training the model. The purpose of phrase combination is to combine different phrases with similar semantics into one phrase. For text generation, long short-term memory (LSTM) and Convolutional neural network (CNN) model are applied. LSTM-CNN model is based on deep learning for training. After training the model, the summary will be generated. LSTM-ATS model encoding, reads the phrase sequence and encodes it into an internal representation and LSTM-ATS model decoding, reads encoded phrase sequence from the encoder and generates the output sequence.

3.3. Flow Chart of the Newly Proposed System

The proposed system aims to develop an Arabic abstractive text summarization system that generates a human-like short summary from long Arabic text. It is very difficult and time consuming for the human to manually summarize large documents of text. Therefore, this system will help in solving this issue by enabling the machine to learn and understand the text and then summarize it into shorter sentences. The developed system uses a deep learning algorithm called Long-Short-Term-Memory (LSTM) to learn and generate the abstractive summary

As shown in Figure 2, at the beginning the user enters the name and password. Then the system checks whether the user is registered on the system or not. After validating the user, the text will be uploaded .The uploaded text will go through the stage of word segmentation followed by the morphological reduction stage and finally the reference resolution. After this phase, the processed text will be displayed to the user and then the process will move through the phrase acquisition followed by phrase refinement and finally phrase combination. After that, a phrase sequence will come out to be applied into the LSTM-ATS model for encoding and decoding. Finally, the system generates text summary and displays it for the user.

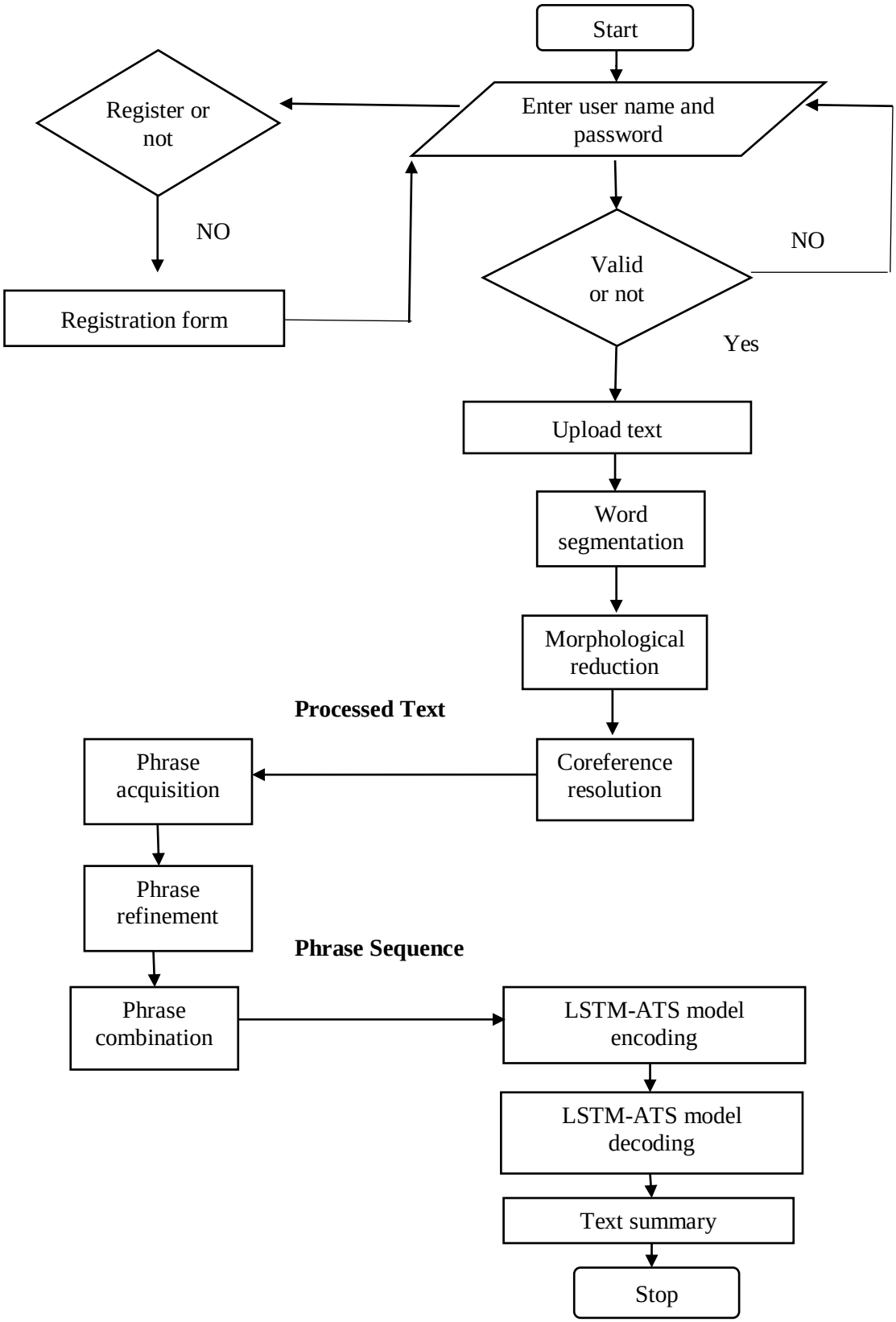

Figure 2. System flow chart

3.3.1. Word segmentation

Word segmentation is the process of dividing the written text into words and determining the sentence words in boundaries and also adding space between words.

3.3.2. Morphological reduction

In case the phrases have same semantics and lose morphology, the morphological reduction will be used for gathering these phrases into one.

3.3.3. Coreference resolution

Coreference resolution is used for identifying or clustering the expressions and noun phrases refer to the same entity in the text. During model training, the ambiguities might be occurred due to the existence of many pronouns in the text that should use coreference resolution before the model is trained to display the processed text.

3.3.4. Phrase acquisition

Phrase acquisition occurs after obtaining the processed text to extract and get the acquired phrases.

3.3.5. Phrase refinement

Phrase refinement is the process of extracting the phrases that should guarantee correct semantics or syntactic structure before the training model.

3.3.6. Phrase combination

Phrase combination is used for improving the redundancy of phrases. Therefore, it will combine different phrases with similar semantics into one phrase and give phrase sequence.

3.3.7. LSTM-ATS model encoding/ decoding

LSTM-CNN model is using deep learning for training. After training the model the summary will be generated..

4. SYSTEM DESIGN

4.1. Hardware Requirements:

- Personal computer. With the following configuration

1. Processor: The proposed model will use (Intel(R) Core (TM) i7-8550U).This is latest technology which will help us to develop the proposed system.
2. Random-access memory (RAM): The proposed model is using RAM (8.00GB).
3. Hard disk: The proposed system needs hard disk with capacity (224 GB).

4.2. Software Requirements

- The Operating system (OS): Windows 10 is the operating system used with proposed system is [12].

- Python 3.0: is a recent, general purpose and higher level programming language. It is available for free, and runs on every current platform .It is also a dynamic object oriented programming language that can be used for many kinds of software development. It offers strong support for integration with other languages and tools [13].

- Google Colab: Colaboratory is a free Jupyter notebook environment provided by Google that requires no setup and it runs entirely under the cloud. With Colaboratory you can write and execute code, save and share the analysis, and also access the powerful computing resources. In addition, Google Colab provides inbuilt version controlling system using Git and it is quite easy to create notes and documentations, including figures, and tables using markdown. It also runs on Google servers using virtual machines [14].

5. IMPLEMENTATION AND TESTING

The implementation is done using Google colab and Python programming language. The implementation of the code is based on four steps as shown in figure 3.

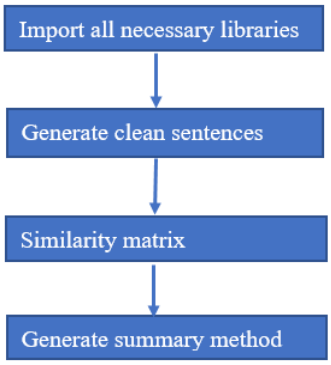

Figure 3 Implemenation steps

5.1. Import all necessary libraries

At first step, all libraries will be called like: **Nltk**, **Corpus**, **Cluster. Util,** and **Cosine_distance**. **Nltk,** is a library in python, **Corpus library is used to** delete the point, comma, etc., to assist in the implementation.

Cluster. Util, is used to call the **Cosine_distance** function which measures the percentage of similarities in sentences as depicted in the code segment shown in figure 4.

```
import nltk
from nltk.corpus import stopwords
from nltk.cluster.util import cosine_distance
import numpy as np
import networkx as nx
```

Figure 4 Importing all necessary libraries

5.2. Inserting the Article

Figure 5 shows, the article title that will be entered to the system.

```
article_url = 'https://sabq.org/kRzGHX'
title = 'تبديل المواليد بين الحماية وسوء الإدارة'
article='
```

Figure 5 Insertion of the article

5.3. Generation of Clean Sentences

Figure 6 **shows** the code segment that reads the article, replaces the comma, places the point, and replaces (..) with (.) ,in addition to separating each sentence with a point.

```
def read_article(article):
    article = article.replace("..", " . ").replace(". ", " . ").replace(" .", " . ")
   article = article.split(" . ")
   sentences = []
   for sentence in article:
   if sentence == '': continue
    print(sentence)
   sentence = sentence.replace("..", " . ")
   sentences.append (sentence.replace (".", " . ").split(" "))
    #sentences.pop()
    return sentences
```

Figure 6 Generation of clean sentences

5.4. Similarity Matrix

Cosine similarity is used to find similarity between sentences, as shown in the code segment depicted in figure 7.

```
def sentence_similarity(sent1, sent2, stopwords=None):
        if stopwords is None:
            stopwords = []
      #sent1 = [w.lower() for w in sent1]
      #sent2 = [w.lower() for w in sent2]
      all_words = list(set(sent1 + sent2))
      vector1 = [0] * len(all_words)
      vector2 = [0] * len(all_words)
      # build the vector for the first sentence
      for w in sent1:
      if w in stopwords:
      continue
     vector1[all_words.index(w)] += 1
  # build the vector for the second sentence
    for w in sent2:
     if w in stopwords:
      continue
     vector2[all_words.index(w)] += 1
    return 1 - cosine_distance(vector1, vector2)
```

Figure 7 Similarity matrix

5.5. Generating Summary Method

Method will keep calling all other help functions to keep the summarization pipeline going as depicted in the code segments shown in figure 8 and figure 9.

```
def build_similarity_matrix(sentences, stop_words):
            # Create an empty similarity matrix
            similarity_matrix = np.zeros((len(sentences), len(sentences)))
              for idx1 in range(len(sentences)):
          for idx2 in range(len(sentences)):
           if idx1 == idx2: #ignore if both are same sentences
            continue
          similarity_matrix[idx1][idx2] = sentence_similarity(sentences[idx1], sentences[idx2], st
           op_words)
            return similarity_matrix
```

Figure 8 Generating summary

```
def generate_summary(file_name, top_n=5):
      nltk.download("stopwords")
       stop_words = stopwords.words('arabic')
      summarize_text = []

       # Step 1 - Read text anc split it
        sentences =  read_article(file_name)

         # Step 2 - Generate Similary Martix across sentences
           sentence_similarity_martix = build_similarity_matrix(sentences, stop_words)

        # Step 3 - Rank sentences in similarity martix
      sentence_similarity_graph = nx.from_numpy_array(sentence_similarity_martix)
      scores = nx.pagerank(sentence_similarity_graph)

        # Step 4 - Sort the rank and pick top sentences
       ranked_sentence = sorted(((scores[i],s) for i,s in enumerate(sentences)), reverse=True)
       print("\n")
      print("Indexes of top ranked_sentence order are ", ranked_sentence)

      print(ranked_sentence)
      for i in range(top_n):
      summarize_text.append(" ".join(ranked_sentence[i][1]))

     # Step 5 - Offcourse, output the summarize texr
      print("\n")
      print("Summarize Text: \n", ". ".join(summarize_text))
```

Figure 9 Code for generating the summary method

5.6. Summarizing the Text

Figure 10 shows the article summary in one sentence and also ranks for each sentence.

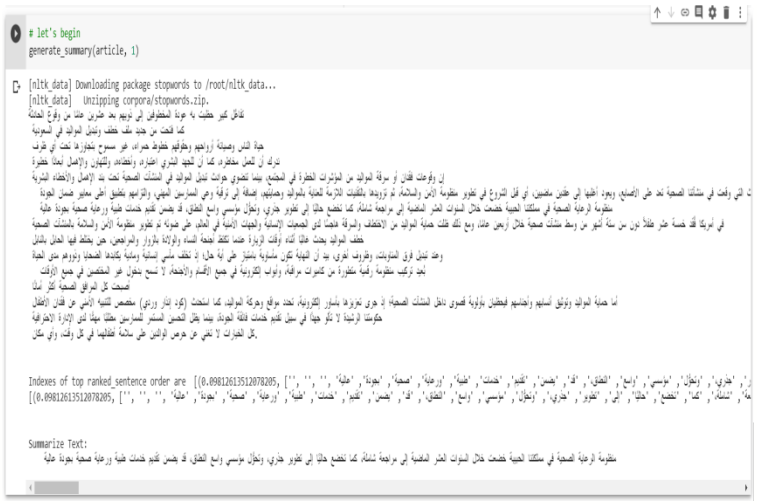

Figure 10 Summarized text

5.7. Implantation Stage

There are three stages namely: sign up page, sign in page, summary page.

5.7.1. Sign up page

The user has to enter his credentials, first name, last name, mobile number, e-mail, password, and confirm password for sign up .

5.8. Sign in page

The user has to enter the email and password to confirm sign in ,in order to upload the article.

5.9. Summary page

In figure 11, the user enters the text and determine the number of sentences, then click Summarize "لخص". As shown in figure 12, it will display the summarization of the text (the number of sentences that was decided by the user) .

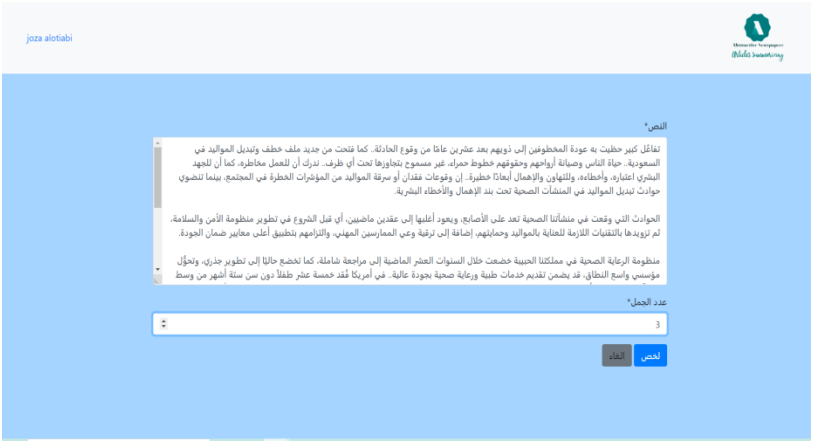

Figure 11 Summary page

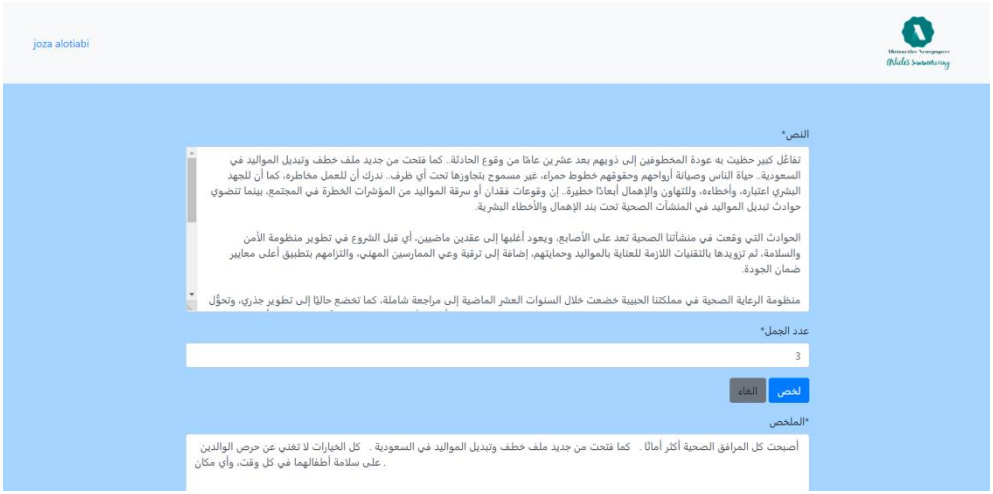

Figure 12 Displaying summary

6. CONCLUSION

Text summarization is the problem of creating a short, accurate, and fluent summary of a longer text document. Automatic text summarization methods are greatly needed to address the ever-growing amount of text data available online for both better help to discover relevant information and to consume relevant information faster. This paper introduced an overview of various past researches and studies in the field of Automatic Text Summarization. The proposed system aims at developing an Arabic abstractive text summarization system that generates a human-like short summary from long Arabic text. It is very difficult and time consuming for the human to manually summarize large documents of text. Therefore, this system will help in solving this issue by enabling the machine to learn, understand and summarize the text into shorter sentences. The developed system uses a deep learning algorithm called Long-Short-Term-Memory (LSTM) to learn and generate the abstractive summary. The implementation of this project had achieved a great result by suggesting a useful summary.

The Future Work will be dedicated to implement the newly proposed system for the newspapers articles summarizing using deep learning approach. More specifically, semantic analysis approach will be used to get deep analysis

REFERENCES

[1] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B.Gutierrez, Krys Kochut, "Text Summarization Techniques: A Brief Survey", arXiv preprint, ArXiv: 1707.02268, (2017).

[2] 1Yogan Jaya Kumar, 1Ong Sing Goh, 1Halizah Basiron, 1Ngo Hea Choon and 2Puspalata C Suppiah, "A Review on Automatic Text Summarization Approaches ", Journal of Computer Science, Vol 178.190, 180-186, (2016).

[3] S.A.Babar, M.Tech-CSE, RIT." Text Summarization: An Overview ", Research Gate.(2013)

[4] Aishwarya Padmakumar, Akanksha Saran, "Unsupervised Text Summarization using Sentence Embeddings ", Semantic Scholar, (2016).

[5] Dr. Michael J. Garbade," A Quick Introduction to Text Summarization in Machine Learning " Towards Data Science, (2018).

[6] Chen Y., Ma Y., Mao X., Li Q, "Abstractive Summarization with the Aid of Extractive Summarization ", Springer, vol 10987, 4-13, (2018).

[7] Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, Noah A. Smith, "Toward Abstractive Summarization Using Semantic Representations ", arXivpreprint, arXiv: 1805.1039, (2018).

[8] Wan X., Li C., Wang R., Xiao D., Shi C," Abstractive Document Summarization via Bidirectional Decoder ", Springer, vol 11323, 365-367, (2018).

[9] Mahajani A., Pandya V., Maria I., Sharma D," A Comprehensive Survey on Extractive and Abstractive Techniques for Text Summarization ", Springer, vol 904,339-349, (2019).

[10] Narayan, S., Papasarantopoulos, N., Cohen, S.B., Lapata, M.m, "Neural extractive Summarization with side information ", arXiv preprint, arXiv: 1704.04530, (2017).

[11] Neri Mendoza V., Ledeneva Y., García-Hernández R.A, "Abstractive Multi-Document Text Summarization Using a Genetic Algorithm ", Springer, vol 11524, 423-431, (2019).

[12] Kinnary Jangla, "Windows 10 Revealed: The Universal Windows Operating System for PC, Tablets, and Windows Phone ", Apress, (2015).

[13] Reeta Sahoo, Gagan Sahoo, "Computer Science with Python ", Saraswati House Pvt Ltd, (2016).

[14] "Google Colab", [OnLine]. Available: https: //colab.research.google.com.