Data Representation for Deep Learning - Based Arabic Text Summarization Performance Using Python Results

Mohamed Yassin Abdelwahab Yassin[1*], Yazeed Al Moaiad[2]

[1,2]*Faculty of Computer and Information Technology, Al-Madinah International University, Taman Desa Petaling, 57100 Kuala Lumpur, Malaysia;* E-mail: co477@lms.mediu.edu.my

Abstracts: A sequence-to-sequence model is used as the foundation for a suggested abstractive Arabic text summarizing system. Our goal is to create a sequence-to-sequence model by utilizing multiple deep artificial neural networks and determining which one performs the best. The encoder and decoder have been developed using several layers of recurrent neural networks, gated recurrent units, recursive neural networks, convolutional neural networks, long short-term memory, and bidirectional long short-term memory. We are re-implementing the fundamental summarization model in this study, which uses the sequence-to-sequence framework. Using a Google Colab Jupiter notebook that runs smoothly, we have constructed these models using the Keras library. The results further demonstrate that one of the key techniques that has led to breakthrough performance with deep neural networks is the use of Gensim for word embeddings over other text representations by abstractive summarization models, along with FastText, a library for efficient learning of word representations and sentence classification.

Keywords: Machine Learning, Sequence-to-sequence, Deep Learning Model, Natural Language Processing, Arabic Text Summarization.

1. INTRODUCTION

Text summarization is generally understood to be the process of utilizing tool for creating a short text from a lengthy text document, where the short text serves as a summary of the main points of the original document [1]. Based on the three types of approaches, text summarization can be divided into categories. Based on the type of input, the first methodology classifies summarization processes as either single-document summarizations or multi-document summarizations.

Single-document summarizing uses a single document as the input, and the summary is created from this document, but multi-document summarizing uses many texts as the input, and the summary should include data from each of these documents.

The second methodological type is context-based, and it divides the summary process into generic, query-driven, and domain-specific summaries.

The last and most important sort of methodology for summarizing text is based on the output type. there are two types: extractive and abstractive summarization. Extractive summarizing builds the summary from sentences or phrases in the source document(s) based on statistical and linguistic variables, as opposed to abstractive summarization, which depicts the ideas in the source documents using various terminology based on the actual semantics of the text. [2]; [3]. Additionally, abstractive summarization involves semantic analysis of the text, which can be accomplished by employing machine learning techniques and sophisticated natural language processing, making abstractive summarization more difficult than extractive summarization [4].However, abstractive summarization is preferable since it is more intelligible and is akin to a summary that is written by humans [5].

Recently, deep learning methods have provided significant improvements in important tasks like text translation [6]; sentiment analysis [7]; and text summarization and others fields [8]. Also, the important feature of using deep neural networks is it takes advantage of big datasets to improve their results [9]. The sequence-to-sequence structure of the encoder-decoder model serves as the foundation for the new text-summarizing techniques. The encoder and decoder are the two components of this design.

Every time a time step occurs, the encoder receives a fresh token from the input sequence and modifies the hidden states in accordance with this token. Regardless of the length of the input, the encoder creates the context vector with a defined length after reaching the final token of the input sequence as the representation of the input. The suggested system was developed using a number of deep artificial neural networks, including RNN, GRU, LSTM, BiLSTM, TNN, CNN, FastText, and Genism, to determine which of them performed the best.

2. LITERATURE REVIEW

Automated text summarizing is a method for producing a summarized document from a connected set of documents by automatically extracting important information from related documents [10, 11]. Text summarization can help in extracting important sentences from various related documents because the amount of text data is currently growing quickly in areas like news, official documents, and medical reports. As a result, it is necessary to compress such data using machine learning techniques [12]. The main issues with document summaries include repetition, noise in the content, incoherence, and reduced readability [13]. One of those natural language processing applications, text summarization is certain to have a significant impact on our daily lives. Natural language processing is a technique that involves extracting key sentences from connected sources. One such technique is ATS. The Text Analysis Conference (TAC) and the Document Understanding Conference (DUC) have seen a large number of studies on English and European languages, but few studies on the Arabic language [14]. There are numerous categories in which on the output type and summarization might be placed. Our search looked at multi-document text summarizing based on sifting out pertinent information from Arabic texts in a broad context [15]. This study's major objective is to present a model for text-summarizing documents that is specific to a certain domain and to use it to produce summary sentences that are extremely relevant to the domain. Text summarization is the process of creating a concise and coherent summary of a longer text while maintaining its main ideas and salient details. Additionally, it captures three crucial characteristics of automatic summarization. In the beginning, summaries are taken from a single text or a collection of documents. Second, important information should be kept in summaries. Finally, they ought to be brief. Figure 1 below demonstrates how text summarizing can be broken down into summaries based on the type of input, the type of output, and the purpose of the summary [16].

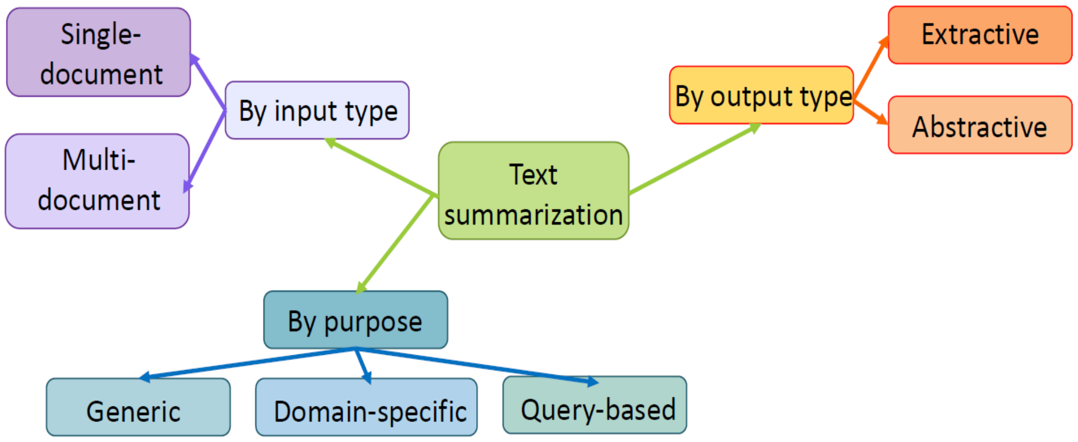


Figure 1. Process of Text Summarization Taxonomy

Text summarization is one of the most important applications of natural language processing (NLP). It is an essential tool for assisting and interpreting text information. The goal of automatic text summarization is to abbreviate one or more texts into shorter version conserving their information contents and overall meanings.

This will help the reader to decide if a document covers the desired information with minimum effort and time loss.

Intelligent applications like Automated Text Summarization (ATS) can be developed to access this information. Specialized Arabic ATS approaches are needed to overcome this problem and to support Arabic Natural Language Processing (NLP) systems.

Text summarization can be categorized into many types. According to input factors, can generate single-document summary from a single input text document, or multi-document summary that is generated from multiple input documents concerning the same topic. With regard to the output of the text summarization, a summarization could be extractive, that selects important sentences from the original document and concatenates them into a shorter form; or, it could be Abstractive, which attempts to understand the main concepts in the document and to express them in a clear language through representing them using new terms, expressing them using new formulations, and mentioning other concepts and words than those in the original text [16].

Deep learning, an area in machine learning, has performed with state-of-the-art results for common NLP tasks such as Named Entity Recognition (NER), Part of Speech (POS) tagging, or sentiment analysis. In case of text summarization, the two common approaches are extractive and abstractive summarization. Sequence-to-sequence learning (seq2seq) has made abstract summarization possible. It has been used successfully in NLP applications such as machine translation, speech recognition, and conversation systems [16].

Deep learning still has concerns with

[1] Creating repeated words or phrases and

[2] Not dealing with terms that are not in the vocabulary.

RNN models use attention encoders and decoders to summarize material effectively.

Deep learning approaches still suffer from challenges such as:

[1] Creating repeated words or phrases and

[2] The inability to cope with words out of vocabulary (OOV) (i.e., unusual and limited occurrence words).

The summarizing mechanism of goes like this:

[1] Separate the actual items (for example, news reports) and their summaries.

[2] After preparing the data with a sub-word display, do word segmentation.

[3] Using a pre-trained Genism toolkit to initialize the word vectors, with one layer of Bilstm for the encoder and a unidirectional.

LSTM layer for the decoder. The cost function was optimized (loss).

Deep learning models are typically employed for brief text summaries.

Combining multiple approaches and tactics are advised to develop improved abstractive summaries.

It is very promising to combine results from numerous ATS techniques to provide considerably better summaries than those created by individual algorithms. These methods might be employed in pre-processing to extract key terms from the input text and then utilized to generate the abstractive summary.

3. METHODOLOGY

This section outlines the various elements and procedures that our system has employed. The Arabic text summary has been the subject of numerous studies. The following subsections describe the various methodologies and algorithms that these researchers used. There are two types of summarization models: extractive and abstractive.

A type of extractive summarization produces extracts by choosing crucial sentences, while another model produces abstracts by comprehending the meaning of the entire text. The extractive method, or those that choose sentences from the original material to construct the summary.

Based on a sequence-to-sequence model, a method for abstractively summarizing Arabic text is proposed in this research. Encoder and decoder are the two elements that make this model function. Our goal is to construct the sequence to sequence model utilizing a variety of deep neural networks and to determine which one performs the best.

Different layers of Recurrent Neural Network (RNN), Gated Recurrent Units (GRU), Recursive Neural Network (Rec NN\TNN), Convolutional Neural Network (CNN), The encoder and the decoder were developed using long short-term memory (LSTM) and bidirectional long short-term memory (BiLSTM). In addition, Deep neural networks use large datasets to enhance their performance. In this study, we reapply the basic summarizing model that makes use of the sequence-to-sequence framework in Arabic, a language that has never before seen the usage of this model in text summarization.

The Keras library, which we used to build these models allows them to operate smoothly on Google Colab Jupiter notebooks. Our suggested system also performs better than the other most recent study investigations. Aside from that, the outcomes demonstrate that abstract summarization models, which chooses to employ word embeddings rather than other text representations in FastText, a toolkit for effective learning of word representations and sentence categorization, is one of the main techniques that has enabled deep neural networks to achieve ground-breaking performance.

3.1. RNN: Recurrent Neural Network Method Performance

An RNN often depends on both current and past occurrences. On the other hand, there are circumstances in which a forecast is dependent upon past, present, and future occurrences. For instance, forecasting which word will appear in a sentence may require us to gaze into the future; that is, a word's placement in a sentence may depend on an upcoming occurrence. These linguistic relationships are prevalent in a number of tasks involving text prediction. Thus, it is useful to record and examine both past and future events. BRNNs, or bidirectional RNNs, are used to permit both forward (future) and backward (past) traversal of input. A BRNN is an aggregation of two RNNs, one of which starts at the beginning of the data series and goes forward, while the other starts at the end and goes backward. the outputs of the two RNNs are typically concatenated at each time step. Depending on the application, a BRNN's individual network blocks can be an LSTM, GRU, or a conventional RNN.

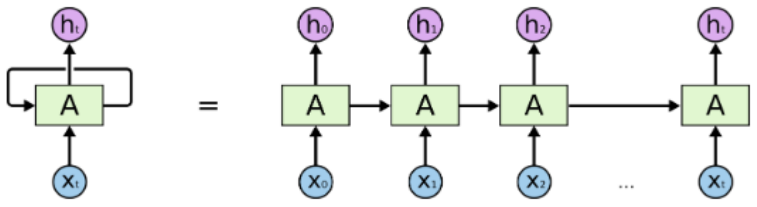


Figure 2. An unrolled recurrent neural network

Where the network uses the output as an input again.

Traditional neural networks do not remember any prior work when they construct an understanding of the job from the examples provided. The word order in the input documents, however, is crucial for jobs like text summarization. When processing the next phrase, we want the model to keep in mind the ones it processed before.

Recurrent neural network (RNN) resembles an unrolled log in appearance. The symbols in the diagram represent the input units for each timestamp, whereas "x" represents the value of the output units following each timestamp, where each timestamp may represent the processing of a single word if the input is a string collection. and "A" stands for a section of the neural network.

The result from the previous timestamp is transferred to the following phase for a component of the calculation that occurs in a sector of the neural network. As a result, the data is obtained from the prior timestamp. The increasing distance between the connected pieces of information, however, makes it difficult for traditional RNNs to memorize information effectively in reality. Considering that each activation function is nonlinear, it is challenging to obtain information by going back hundreds or thousands of processes.

3.2. LSTM: Long Short-Term Memory Method Performance

Another type of recurrent neural network that may recognize long-term dependencies is called an LSTM. An LSTM block performs a few more functions than an RNN, which has a single layer in a network block. It retains the important information it learns throughout the network and discards the irrelevant information by using input, forget, and output gates.

- Forget Gate discovers which information should be removed from the block.
- The input gate determines which input value should be applied to change the memory.
- The input is detected by the output gate, and the output is determined by the block's memory.

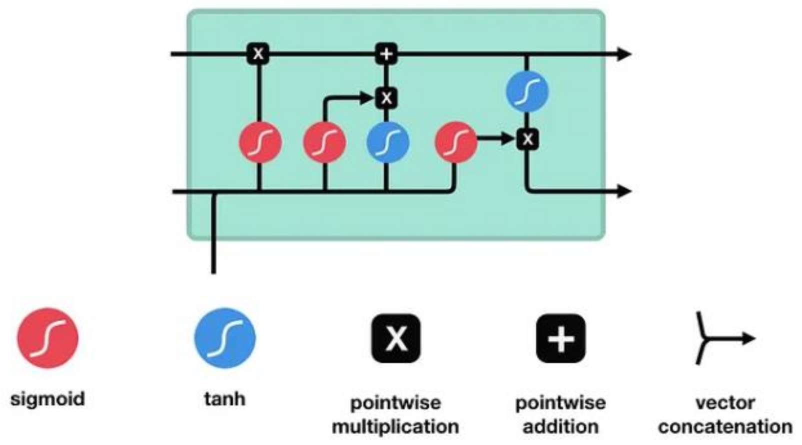


Figure 3. Long-Short Term Memory (LSTM) Neural Network

Long Short-Term Memory (LSTM) networks are capable of long-term information transfer. In contrast to the conventional RNN, each LSTM cell contains a number of straightforward linear operations that enable data to be transmitted without requiring complicated computation. By performing some linear operations, the prior cell state containing all the information up to that point smoothly traverses an LSTM cell.

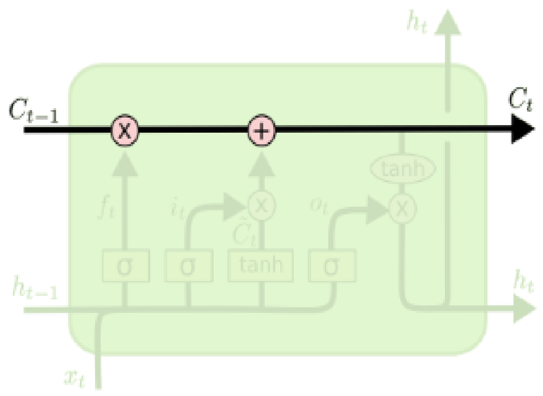


Figure 4. The Linear operations in an LSTM cell

Each LTSM cell within selects what data to keep and when to allow reads, writes, and erasures of data using three gates that open and close. Figure 5 labels the first gate as the "forget gate layer," which receives the value of the previous output units, ht-1, and the current input, xt, and outputs a number between 0 and 1, denoting the amount of information going through. All information is allowed to pass when the value is 1, whereas no information is allowed to pass when the value is 0.

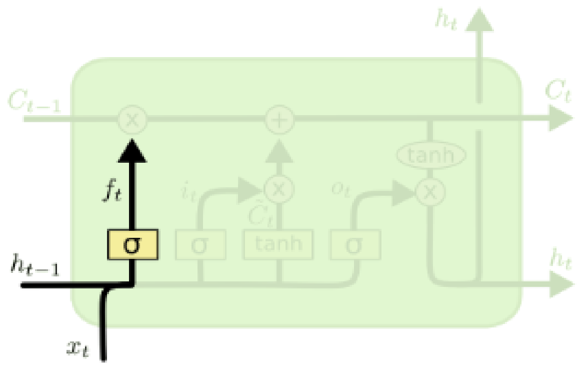


Figure 5. "The Input gate Layer"

What information needed to be modified is decided using the "input gate layer" of the LSTM. It accepts the value of the previous output units, ht-1, in addition to the current input xt, and outputs a number that indicates which cells should have the information updated. The previous cell state, Ct-1, is then switched out for the new state, Ct.

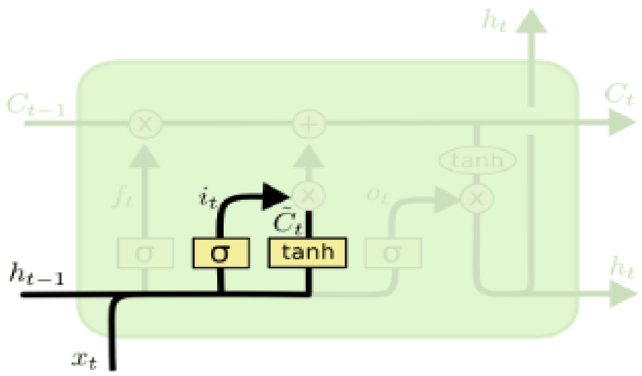


Figure 6. "The forget gate layer"

The last gate, also referred to as the "output gate layer," defines what the outcome should be. The weighted output of the sigmoid function is compounded with the cell state, as shown in Figure 7, after being processed by a tanh function in the output layer. As a result, the following LSTM cell receives the value of the output units, ht.

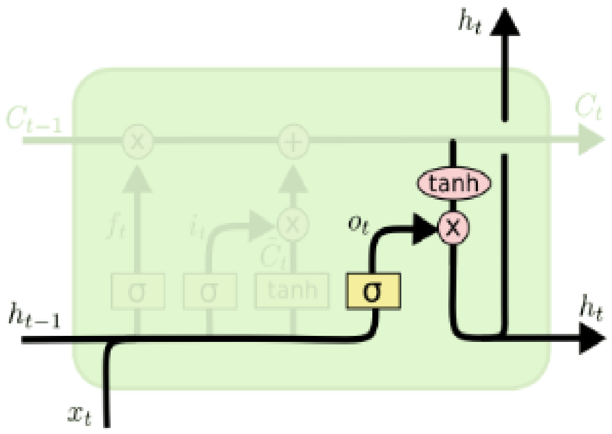


Figure 7. "The Output gate layer"

The three gate levels are connected by basic linear operators. The massive LSTM neural network is made up of numerous LSTM cells, and regardless of how many cells the network has, all information is passed through each cell while the crucial information is retained until the conclusion.

3.3. GRU: Gated Recurrent Unit Method Performance

The Gated Recurring Unit, which debuted after RNN and LSTM, provides an advantage over the other two dues to its recent entry. The GRU, often referred to as a gated recurrent unit, is superior to the conventional RNN, or recurrent neural network. LSM and GRUs share a lot of similarities (LSTM). The GRU regulates the information movement using gates, just like the LSTM. In comparison to LSTM, they are still reasonably new. Because of this, they have a more straightforward design and provide some improvements over LSTM because of this.

It requires an input Xt and the hidden state Ht-1 from the timestamp prior to that at each timestamp t. The next timestamp is once more conveyed with the use of a new hidden state called Ht. A GRU has two gates at the moment, as opposed to an LSTM cell's three gates. The first gate is referred to as Reset, while the second gate is referred to as Update.

Reset Gate (Short term memory)

The network's short-term memory, also known as the hidden state, is controlled by the reset gate (Ht). This is how the Reset gate's calculation looks.

$$r = \sigma(X_t * U_r + H_{t-1} * W_r) \quad (1)$$

LSTM gate formula That and it are very comparable. Rt will be given a value between 0 and 1 by the sigmoid algorithm. The Ur and Wr symbols in this diagram represent the reset gate's weight arrays.

Update Gate (Long Term memory)

Similarly, we have an Update gate for long-term memory and the equation of the gate is shown below.

$$U_t = \sigma(X_t * U_u + H_{t-1} * W_u) \quad (2)$$

Uu and Wu are weight measures that differ only slightly from one another.

How GRU Works

Now let's look at the operation of these gates. For the hidden state Ht in GRU to be discovered, two steps must be taken. The first phase consists of creating the candidate's concealed state. Like what is seen below Candidate Hidden State.

$$H^{\wedge}_t = \tanh(X_t * U_g + (r_t \odot H_{t-1}) * W_g) \quad (3)$$

The input is multiplied by the reset gate output rt and given the concealed state from the previous timestamp, t-1. The resultant value of the tanh function after receiving this information later on serves as a representation of the candidate's hidden state.

The most crucial aspect of this equation is the ability to regulate the amount of influence that the prior concealed state can have on the candidate state by adjusting the reset gate value. If rt has a value of 1, then all of the data from the preceding concealed state, Ht-1, is being considered. Information from the prior concealed state is completely disregarded if rt is equal to 0.

Hidden state

When the candidate state is acquired, it is then used to create the current concealed state Ht. At this point, the Update gate appears on the scene. Instead of employing separate gates like in LSTM, this equation uses a single gate, we use one update gate in GRU to control both the fresh information from the candidate state and the historical information, which is represented by Ht-1.

$$H_t = U_t \circ H_{t-1} + (1 - U_t) \circ H^{\wedge}_t \quad (4)$$

The first term in the equation will vanish if the value of ut is close to 0, which indicates that not much information from the previous hidden state will be included in the current hidden state. Nevertheless, the second part almost completely combines with the first, which suggests that the candidate state will be the only source of information for the hidden state at the present timestamp. The current hidden state will solely depend on the first term or the information from the hidden state at the earlier timestamp t-1 if the value of ut is on, similarly to the previous example. The second term will completely become 0 if the value of ut is on. It follows that the value of ut, which ranges from 0 to 1, is crucial to understanding this equation.

End Notes

LSTM has three gates whereas GRU only has two. They are called the Input, Ignore, and Output gates in LSTM. While the GRU has a reset gate and an update gate. The LSTM consists of two zones. Short-term memory is sometimes referred to as hidden state, whereas long-term memory is also referred to as cell state. The hidden state (Ht) is the only state present in the GRU instance.

The GRU contains an update and a reset gate to control the path of data entering the network and decide which information is kept and which is forgotten. The update gate determines whether the cell will modify its values through the incoming data or not, and its equation is:

$$\mathbf{Z}_t = \boldsymbol{\sigma}(\mathbf{W}^{(z)}\mathbf{X}_t + \mathbf{U}^{(z)}\mathbf{h}_{t-1}) \quad (5)$$

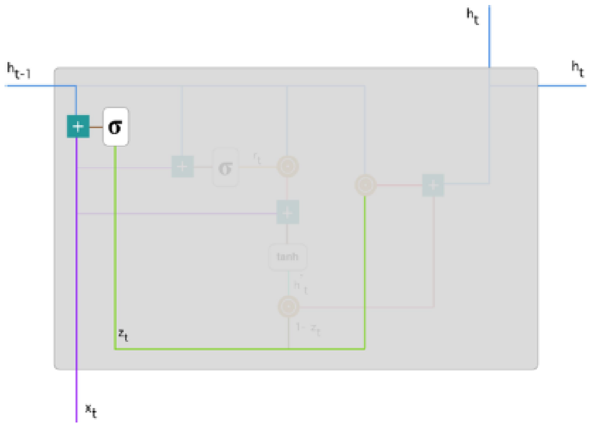


FIGURE 8. UPDATE AND A RESET GATE TO CONTROL THE PATH OF DATA ENTERING

Where x is the input, w is its associated weight, h is the output of the previous cell, and u is its associated weight. As for the reset gate, it determines whether the previous cell is significant or not, and sometimes it is not used in GRU, and its formula is:

$$r_t = \sigma \left(W^{(r)} X_t + U^{(r)} h_{t-1} \right) \qquad (6)$$

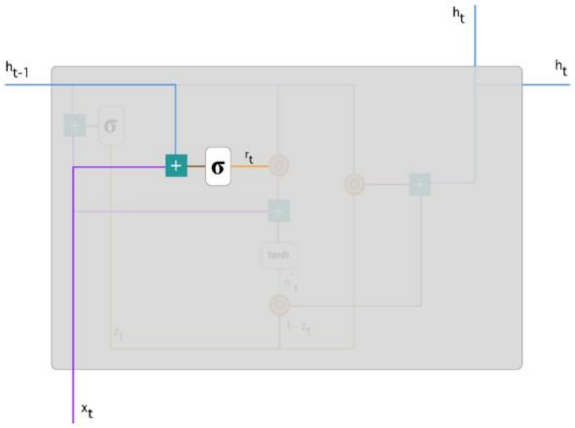


Figure 9. reset gate determines whether the previous cell is significant or not

Where x is the input, and w is the weight associated with it, which is different from the weight of the previous gate, and h is the output of the previous cell, and u is the accompanying weight, which is also a different weight.

Then step the tanh by multiplying x by its weight, then adding it to the product of multiplying the output of the reset gate by h multiplied by its weight.

$$\acute{h}_t = \tanh(WX_t + r_t \odot Uh_{t-1} \qquad (7)$$

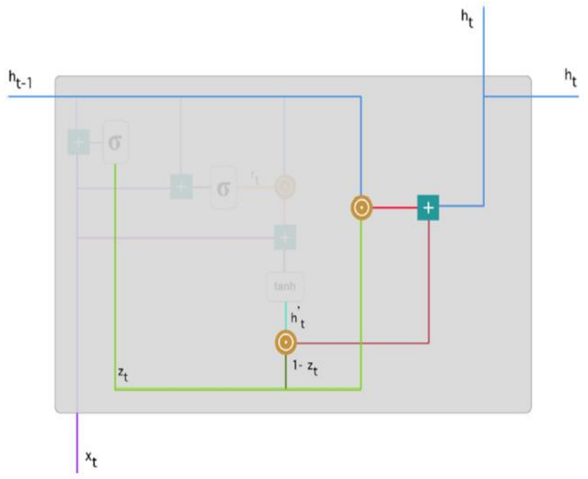


Figure 10. calculating the tanh equation reset gate

Finally, the final product by multiplying the output of the update gate by the value of the previous cell, set to 1 minus the value of update multiplied by the current default cell value.

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \acute{h}_t \qquad (8)$$

In GRU architecture, there are two gates: reset and update; however, in LSTM architecture, there are three gates: input, output, and forget. This is the main distinction between them. Due to the fact that GRU has fewer gates than LSTM, it is less complex. For this reason, GRU is favored for small datasets, while LSTM is favored for bigger ones.

3.4. Rec NN\TNN: Recursive Neural Network Method Performance

It is a type of neural network that is mainly used with texts, and that has a hierarchical shape different from the traditional one, and sometimes it is called Tree Neural Network and stands for TNN. One of the problems facing the bag of words model is that the order is very important, as the words dog toy and toy dog have different meanings.

- "أنا أحب الرياضيات وأكره الكيمياء"
- "أنا أحب الكيمياء وأكره الرياضيات"

Both will have the same values in the Bag of Words (BOW) table while they are different in meaning.

It is intended to determine the extent to which each word is present in the text, and the extent of its repetition, which may refer to a specific meaning of this text.

This matrix can be used to find out the approximate meaning of the sentence, or to classify it and determine whether it is appropriate or not.

Disadvantages are that it does not focus on the meanings of the words, and that the process is very slow.

Table 1: BOW Value Different Meaning

	أنا	أحب	الرياضيات	و	أكره	الكيمياء
Phrase1	1	1	1	1	1	1
Phrase2	1	1	1	1	1	1

Also, RNN has a problem with large sentences, which are unable to link the required word with a specified number of previous words. Human, by nature, does not read the whole large sentence at once, but divides it into small parts in order to understand it.

If have a comment on a sentence, and we want to know whether this comment is negative or positive, then the hierarchical form can extract the meaning of each word and each sentence separately, to find out whether the overall meaning is positive or negative.

We see that there are words and parts with a negative red meaning and positive meanings blue and others with a number 0, i.e. neutral, and the final outcome of the sentence can be drawn that it was positive with a number so-and-so.

With the emphasis that only the first half of the sentence There are slow and repetitive parts has a clear negative meaning, but linking it to the second part erased the negative meaning and gave it a positive meaning.

For example: "قصيدة الذبياني"

"ولا عيب فيهم غير أن سيوفهم، بهن فلول من قراع الكتائب"

Also, there are different uses for both RNN and TNN.

- The RNN can be used to deduce the next missing word, such as text generation or question answering applications.

- While TNN has no next word, the words are already in a tree layout, so they can do sentimental analysis of sentences, classify them and see which class they belong to.

- TNN is distinguished from other types of algorithms, in that the rest of the algorithms may not give attention to the word not, which may change the entire meaning of a sentence, while TNN has a great ability to attention to and understand it.

What about its mathematical equation?

The value of a cell is always calculated based on the values of its children nodes, so the cell h1 is its equation

$$h1 = f(W_{left}h_1 + W_{right}h + b) \qquad (9)$$

The higher cells depend on the values of the cells below them

$$h_{root} = f(W_{left}h_1 + W_{right}X_3 + b) \qquad (10)$$

Provided that W is the value of the weight’s matrix and the value of b is the deviation coefficient, and the function is one of the activation functions.

Finally, the Softmax is applied at the end to determine the final value of the sentence, is it positive or negative (or from any category if it is a multiple classification).

$$P(y|h) = \text{softmax}(W_o h + b_o) \quad (11)$$

3.5. Convolutional Neural Network Method Performance

Convolutional neural network model in use has three layers. A convolutional neural network functions in a manner that is fundamentally similar to the visual cortex of an animal brain. Convolutional neural networks provide promising performance in text categorization challenges.

The sole distinction between text classification criteria and image classification criteria is that word vectors are used as a matrix to represent classification results rather than pixel values. Tensorflow, a Python package, is used to implement the suggested model [17].

Figure 11: Illustration of a CNN architecture for sentence classification the model that was used has three layers. The first layer, called the embedding layer, transforms the words into their corresponding embedding vectors.

The second layer, called the convolution layer, is where the model's primary processing happens. A low-dimensional matrix is created by applying predefined filters on the sentence matrix. The SoftMax layer, a down sampling layer that can compute the loss function and reduce the sentence matrix, is the third layer.

The word embedding of the sentence is obtained using the embedding lookup function. Then, padding is applied to the matrix created by the embedding layer to evenly distribute each sentence.

The matrix will then start to be reduced by the defined filters, and convolved features will start to appear. Then, these complicated qualities are further diminished.

The output produced by the convolved features is then dispersed throughout the maximum pooling layer for additional output sampling.

There are specified filters of various sizes and forms. The original sentence matrix will be rolled over by the filters, making it a low-dimensional matrix. We make advantage of TensorFlow's embedding lookup feature rather than training our own embedding. To ensure that all sentence matrices are the same size and shape, embedded sentences are then padded.

The CNN searches for features in the elements, which will try to apply as well with texts. The first step is to convert the texts into an array to be dealt with by CNN, which will be as follows:

1) In the beginning, to get acquainted with the regular data matrices. the rows are for the sample size, while the columns are for the information about it, features.

2) While in the text matrix the rows are for each word separately, and the columns are for the embedding values, and this does not have the same idea as the edges in the images, and therefore applying the idea of filter 2 * 2 that wraps the image will not have a logical meaning.

3) So, a filter will be made that has a width similar to the width of the main table, that is, it will have the same number of embedding values (colored filters), and the wrapping will be vertical only, from top to bottom because the width is the same.

4) The mathematical process is as follows: The filter (brown, which is 5 * 4) is multiplied by all its values in the first four rows of the text matrix and has a specific value, then it moves a step down and the multiplication is done and the value is calculated, and so on, so can find that the output will be values only, which are the number of steps Going down, so there is a 4*1 matrix on the right.

5) And the command is repeated with more than one filter, and each filter has the output matrix of the same color on its right.

6) The filter is designed so that there are those who cover 4 words together (the first and second filter), or cover three words (the third and fourth), or cover two words (the fifth and sixth), and the number of filters is multiple of these numbers, so there may be 50 filters of each design.

7) The next step is to do maxpooling for each category, so that the highest value in each category or color is chosen, and they are stacked together in one vector, and of course will not need the flatten step because all the values are from one dimension.

8) After these values are stacked, they are fed into NN and there are a number of required outputs to be trained on.

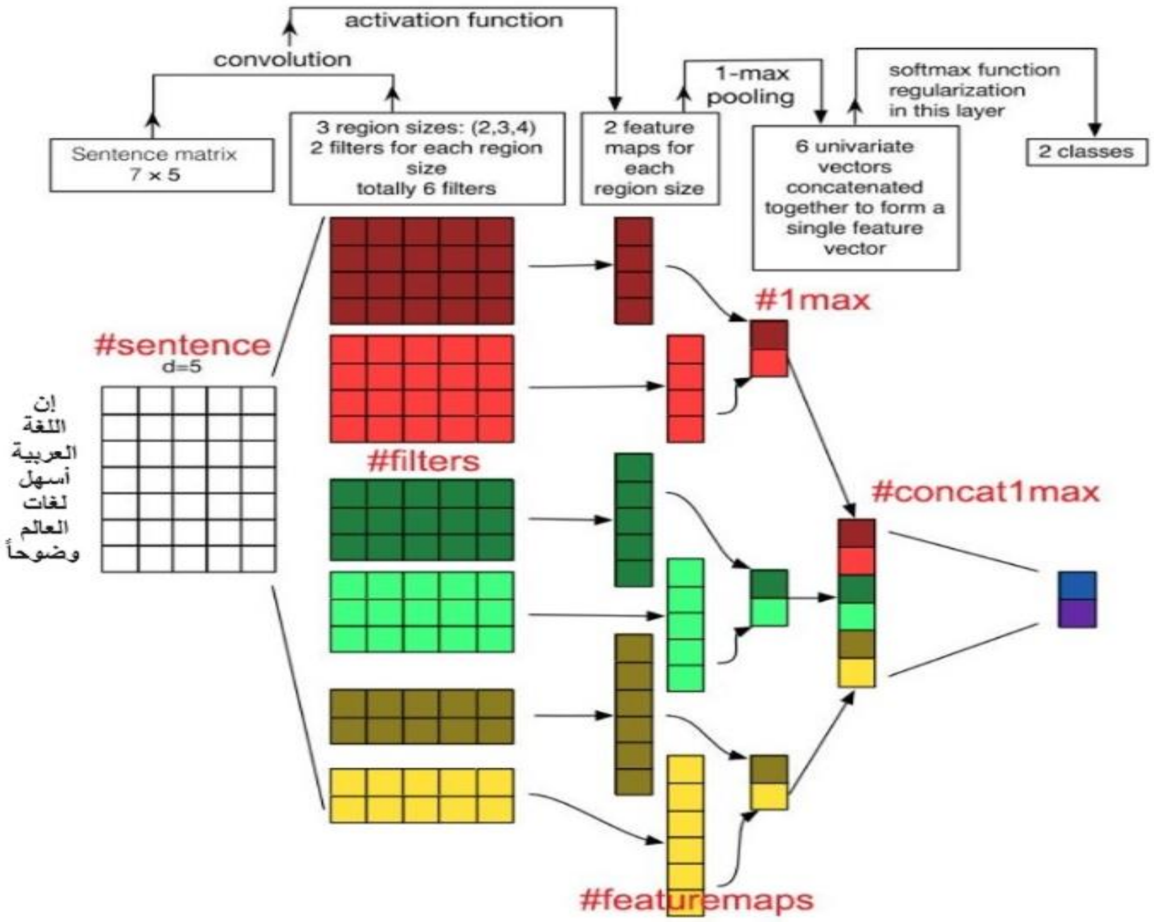

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
#sentence
d=5
إن
اللغة
العربية
أسهل
لغات
العالم
وضوحاً
#filters
#1max
#concat1max
#featuremaps

Figure 11. Text categorization model using a generic CNN architecture with an NLP matrix for 2 classes

3.6. Techniques for Word Embedding and Word-to-Vector

According to the meaning of the term in the document, word embedding, often referred to as word representation, is essential in the generation of continuous word vectors [18].Word embedding is a method for measuring word similarity that is frequently used in NLP applications. It captures both semantic and syntactic information about words.

A two-layer neural network is a tool for handling word2vec in text processing. Natural language is a complex structure that is utilized for expressing meanings.

1. Words act as the fundamental unit of meaning in this system.

2. Word vectors are, as their name implies, vectors that are used to represent words.

3. They can also be considered to be word representations or feature vectors.

4. Word embedding is the process of turning words into real vectors.

5. Word embedding has gradually evolved in recent years into the fundamental understanding of natural language processing.

6. The text corpus is its input, and its output is a large number of arrays containing the text's features.

7. There are two different learning architectures to establish meaning relations.

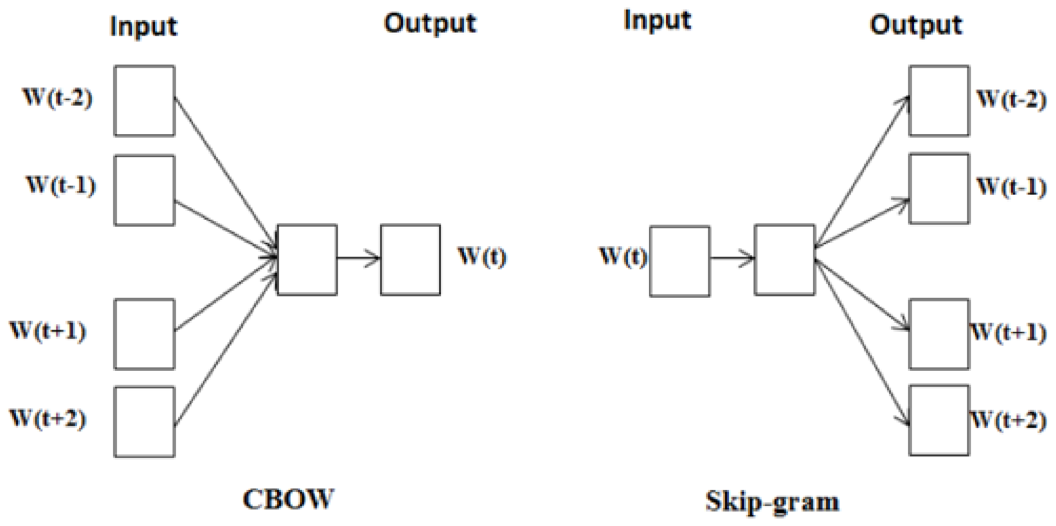


Figure 12. CBOW and Skip-Gram Model Architecture

Table 2: Comparison between CBOW and Skip Gram Tanique

First: CBOW Technique	Second: Skip Gram Technique
It is an abbreviation of continuous bag of words, and it is a mixture between bag of words technique and Ngram technique. The idea of BOW depends on using a number of words in the text and making the numbers 1 and 0 according to the presence of each word. The idea of Ngram depends on the use of one or more of the previous words to infer the next word, and therefore by using CBOW we can use more than one word in the same sentence to infer a specific word, and it can be defined as a neural network, but it is used to predict what is the missing word in a particular sentence (often the last word). This is done by calculating the embedding matrix for the input words, from which they are processed to reach the next layer in the network, and finally the final layer with the SoftMax activation function to select the missing word.	Skip-gram architecture is the alternative approach. Similar to CBOW, this technique operates. Skip-Gram, in contrast to CBOW, predicts the neighbors of the word that is positioned in the middle of the window. This method's benefit is that it can record words' various interpretations, which can be confusing.

3.7. FastText

Developed by Facebook in 2016, FastText is a Word2Vec-based paradigm. The fact that words are broken down into n-grams distinguishes this approach from Word2Vec. In other words, this approach may capture the meaning of closeness that Word2Vec cannot [19].

FastText is a library for efficient learning of word representations and sentence classification. Ex: "Text Classification, Word Similarity, Embedding Values and BOW". These are files that contain hundreds of millions of words that have been trained on Wikipedia, and include more than one language file. English, as well as many other files for 150 languages, including Arabic.

It is also possible to open one of the models that have already been trained on, and use it to find words close to certain words, and it is also possible to deal with the previously trained models, and the models that Facebook has trained on Wikipedia. We can also use fast text from the genism library, as it is fast and easy.

3.8. Gensim

It is an outstanding library in NLP.

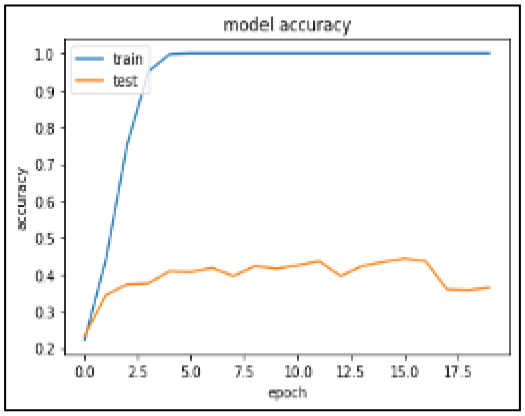
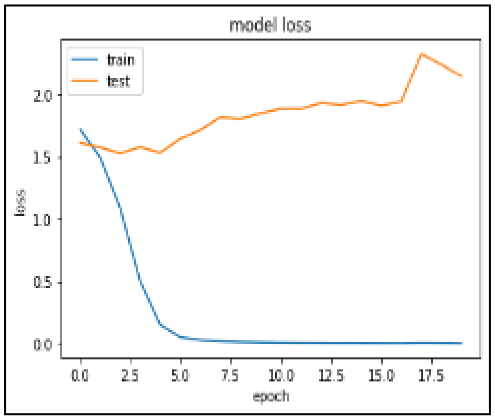
Word embeddings are a modern approach for representing text in natural language processing.

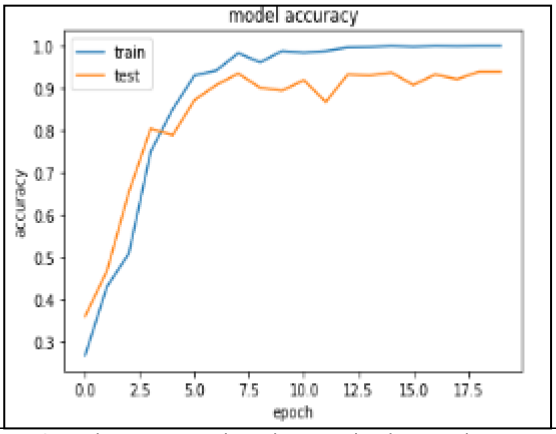
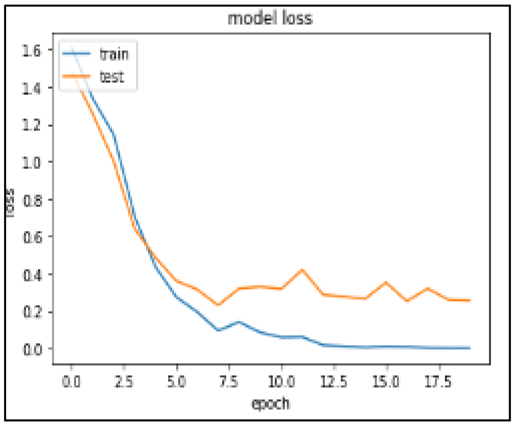
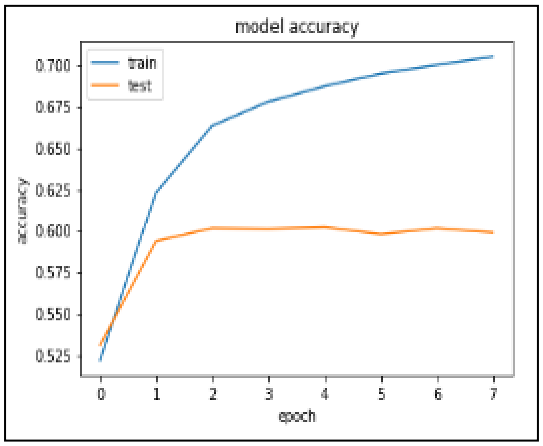
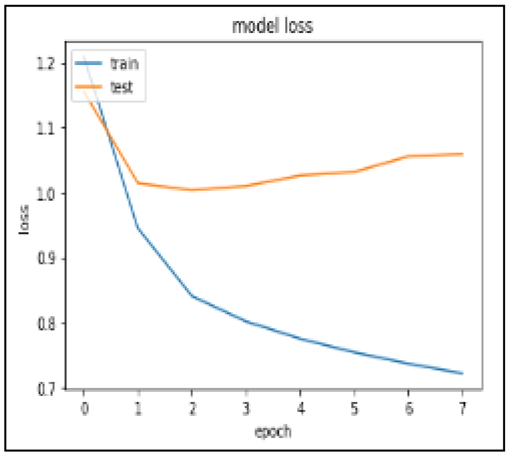
A method for giving words a dense vector representation that conveys some aspect of their meaning is called word embedding. Compared to more basic bag-of-word model word encoding systems such as word counts and frequencies, which produce big and sparse vectors (mainly 0 values) that describe documents but not the meaning of the words, word embeddings are an improvement.

The way word embeddings function is by training a set of dense, continuous-valued, fixed-length vectors using an algorithm on a sizable corpus of text. In the embedding space, every word is represented by a point that is learned and repositioned depending on the words around the target word.

One of the main techniques that has allowed deep neural networks to achieve breakthrough performance on tasks like machine translation is the use of word embeddings over alternative text representations.

Table 3: Representing Data for Arabic NLP Method Performance Tools Results Summary

	Model Accuracy	Model Loss
RNN		
Performance	RNN: one of its shortcomings is that it needs complete data to start processing, and therefore it is not suitable for data that comes simultaneously and continuously, such as converting the incoming voice into texts and (not accuracy). RNN can be used to deduce the next missing word, such as text generation or question answering applications.	

LSTM		
Performance	LSTM: is more complex, slower to implement, but more efficient. LSTM Accuracy: 85.5%	
GRU		
Performance	GRU is simple to configure, it is faster to implement, and it is suitable for deep networks, but its efficiency is not Optimum.	
TNN	• TNN has no next word, the words are already in a tree layout, so they can do sentimental analysis of sentences, classify them and see which class they belong to. • TNN is distinguished from other types of algorithms, in that the rest of the algorithms may not attention to the word not, which may change the entire meaning of a sentence. • TNN has a great ability to attention to and understand it.	
CNN	• provides more accurate test results than LSTM while requiring less training time and around the same weights. As a result, CNN allows for faster training, which decreases the amount of time needed to learn a huge dataset. • CNN using LSTM Accuracy:88.15%.	
Fast Text	An efficient tool for categorizing sentences and representing words is called FastText. Ex: • Text Classification • Word Similarity • Embedding Values • BOW	
Genism	• As it is fast and easy. • One of the fundamental techniques that has enabled deep neural networks to perform at a breakthrough level for challenges like machine translation is the adoption of word embeddings over alternative text representations. • Word embeddings are an improvement over less sophisticated word encoding methods that employ a bag-of-words model, such as word counts and frequencies. • An algorithm that generates a collection of fixed-length, dense, and continuous-valued vectors for word embeddings is trained on a huge corpus of literature.	

CONCLUSIONS

Deep learning approaches, which employ several processing layers to build hierarchical representations of data, have been used to produce modern outcomes in numerous fields. Natural language processing has recently seen a diversity of model designs and techniques emerge (NLP). examined important deep learning-related models and techniques that have been used for a variety of NLP tasks and gave a walk-through of how they developed. Give a full understanding of the past, present, and future of deep learning in NLP along with an overview of the various models and their comparisons and differences. In this research, a sequence-to-sequence-based abstractive Arabic text summarization method has been proposed. The investigation into five areas that affect the quality of the generated summary is the main strength of this paper. We discovered that the optimum performance is achieved by the encoder using six layers of BiLSTM hidden states. the type of deep artificial neural network and the number of its layers utilized to create the encoder and the decoder make up the first direction. The first LSTM results in a superior test accuracy. As a result, CNN allows for faster training, which decreases the amount of time needed to train a big dataset. An efficient tool for categorizing sentences and representing words is called FastText, for example: Text Classification, Word Similarity, Embedding Values and BOW. The second GRU approach is ideal for deep networks, faster to install, and easier to configure, although it is less efficient. The third direction is the method of preprocessing data, and we discovered that the CNN had a crucial impact on getting the best results and offered improved test accuracy compared to LSTM with roughly equal weights and shorter training time. Convolution neural network text classifier implementation is presented in the study. The results showed that the FastText of word representations and sentence categorization produced improved summary quality in the fifth direction, which is the word embedding model that is utilized. The fourth direction of TNN is distinguished from other types of algorithms, in that the rest of the algorithms may not give attention to the word not, which may change the entire meaning of a sentence’s has a great ability to attention to and understand it. The fifth direction of one of the fundamental techniques that has enabled breakthrough performance with deep neural networks is the use of word embeddings over other text representations. This is an improvement over word counts and frequency, which are simpler word encoding approaches for the bag-of-words model. It also explains how to combine the convolution neural network model with the suggested open domain on the basis of deep learning, abstract Arabic text summarization. Genism As it is fast and easy, one of the fundamental techniques that has enabled deep neural networks to perform at a breakthrough level on tasks like machine translation is the adoption of word embeddings over alternative text representations. Word embeddings are an advance over simpler word encoding techniques like word counts and frequencies that use a bag-of-words model. Using an algorithm and a sizable corpus of text, the process by which word embeddings function is to train a set of dense, continuous-valued, fixed-length vectors.

REFERENCES

[1] Al-Saleh, A.B. and M.E.B. Menai, Automatic Arabic text summarization: a survey. Artificial Intelligence Review, 2016. 45: p. 203-234.

[2] Al Qassem, L.M., et al., Automatic Arabic summarization: a survey of methodologies and systems. Procedia Computer Science, 2017. 117: p. 10-18.

[3] Sarkar, K., Using domain knowledge for text summarization in medical domain. International Journal of Recent Trends in Engineering, 2009. 1(1): p. 200.

[4] Azmi, A.M. and N.I. Altmami, An abstractive Arabic text summarizer with user-controlled granularity. Information Processing & Management, 2018. 54(6): p. 903-921.

[5] Sunitha, C., A. Jaya, and A. Ganesh, A study on abstractive summarization techniques in Indian languages. Procedia Computer Science, 2016. 87: p. 25-31.

[6] Bakator, M. and D. Radosav, Deep learning and medical diagnosis: A review of literature. Multimodal Technologies and Interaction, 2018. 2(3): p. 47.

[7] Zhang, L., S. Wang, and B. Liu, Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2018. 8(4): p. e1253.

[8] Liu, H., et al., Adaptive fuzzy synchronization for a class of fractional-order neural networks. Chinese Physics B, 2017. 26(3): p. 030504.

[9] Al-Maleh, M. and S. Desouki, Arabic text summarization using deep learning approach. Journal of Big Data, 2020. 7: p. 1-17.

[10] Abbasi-ghalehtaki, R., H. Khotanlou, and M. Esmaeilpour, Fuzzy evolutionary cellular learning automata model for text summarization. Swarm and Evolutionary Computation, 2016. 30: p. 11-26.

[11] Zou, W.Y., et al. Bilingual word embeddings for phrase-based machine translation. in Proceedings of the 2013 conference on empirical methods in natural language processing. 2013.

[12] Sanchez-Gomez, J.M., M.A. Vega-Rodríguez, and C.J. Perez, Comparison of automatic methods for reducing the Pareto front to a single

solution applied to multi-document text summarization. Knowledge-Based Systems, 2019. 174: p. 123-136.

[13] Verma, P. and H. Om, MCRMR: Maximum coverage and relevancy with minimal redundancy based multi-document summarization. Expert Systems with Applications, 2019. 120: p. 43-56.

[14] Mallick, C., et al., Graph-based text summarization using modified TextRank, in Soft computing in data analytics. 2019, Springer. p. 137-146.

[15] Kanapala, A., S. Pal, and R. Pamula, Text summarization from legal documents: a survey. Artificial Intelligence Review, 2019. 51(3): p. 371-402.

[16] Abdelwahab, M.Y., Y. Al Moaiad, and Z.B.A. Bakar, Arabic Text Summarization using Pre-Processing Methodologies and Techniques. Asia-Pacific Journal of Information Technology & Multimedia, 2023. 12(1).

[17] Zhang, Y. and B. Wallace, A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820, 2015.

[18] Li, Y. and T. Yang, Word embedding for understanding natural language: a survey. Guide to big data applications, 2018: p. 83-104.

[19] Joulin, A., et al., Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759, 2016.

DOI: https://doi.org/10.15379/ijmst.v11i1.3646