Journal of Theoretical and Applied Information Technology
15th July 2025. Vol.103. No.13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPACT OF MINIMUM SPANNING TREE ALGORITHMS ON EXTRACTIVE ARABIC TEXT SUMMARIZATION APPROACH

AKRAM EL KHATIB[1], GAMAL BEHERY[2], REDA ELBAROUGY[3]
[1] Faculty of Technology & Applied Sciences, Al-Quds Open University, Palestinian.
[2] Department of Computer Science, Faculty of Computer and Artificial Intelligence, Damietta University New Damietta, Egypt.
[3] Department of Information Technology, Faculty of Computer and Artificial Intelligence, Damietta University New Damietta, Egypt.
E-mail: [1] akkhatib@qou.edu, [2] gbehery@du.edu.eg, [3] elbarougy@du.edu.eg

ABSTRACT

The purpose of this research is to investigate the impact of using Minimum Spanning Tree (MST) algorithms for enhancing the Arabic Text Summarization (ATS) graph-based approach's performance. The previous researches were conducted in an extractive ATS that relied on a graph approach are very limited, and their performance is still low. This low performance is attributed to the characteristics of Arabic language which is morphologically complex, moreover, there is a lack in ATS researches using graph-based technique. The final results of the graph-based technique mainly rely on the weights between sentences as major features which are poorly calculated. To address these limitations, this study applies and evaluates three MST algorithms (Prim's, Kruskal's, and Bourka's) within a single-document extractive ATS system. The proposed system converts text into a graph where sentences are nodes and similarity-based weights are used as edges. The MST algorithm is then applied to extract the most representative sentences. To ensure objective comparison, the Essex Arabic Summaries Corpus (EASC) was used as a benchmark dataset. Experimental results show that Kruskal's MST algorithm achieves the best performance, demonstrating a significant improvement of 15.2% in recall and 14.3% in F-measure over previous single-document extractive ATS methods. This confirms the effectiveness of MST-based graph algorithms in improving Arabic text summarization quality.

Keywords: *Extractive Arabic Text Summarization, Arabic NLP, Graph Model, Minimum Spanning Tree Algorithm.*

1. INTRODUCTION

Given the rapid advances in communication using the Internet, the offered size of available information has increased significantly. Consequently, it is crucial to build systems that automatically summarize texts. These systems aim to reduce the required time to obtain useful information listed in a specific document intended to facilitate readers finding information of interest in that text. Text summarization systems are an ideal solution to save time and effort. The summary is the retrieved sentences from the document which has the following feature: informative, small, time-saving and computerized [1].

In general, the summarization process is divided into three main stages: analysis, transformation and synthesis. While in the former stage, the texts and their necessary characteristics are explained using: Stemming analysis(reducing words to their base/ root form), morphological analyzer(examining the structure and formation of words), normalization(reshaping words letters), stop-words elimination and procedures of features extraction, in the transformation stage, sentences are represented as a graph(a structure consisting of nodes and edges) using Minimum Spanning Tree (MST) algorithms(graph algorithms that connect all nodes with minimum total edge weights without forming cycles). In addition, the final stage in which the final summary is constructed.

Three factors influence text summarization approaches as per the following: the number of documents to be summarized, the document's goal and type [2]. and whether the summary is query-focused or question-based [2]. In addition, in conformity with the type of summary sentences summarization systems could be divided into two groups: summary of extractive or abstractive

summary. The extractive texts approach chooses the main unmodified sentences, to get the summary. Also, abstractive summary requires techniques to interpret, understand and analyze the original texts [2][3].

Arabic has a unique value as it is spelled from the right. In addition, most Arabic words start with a constant letter. Arabic has three vowels that distinguish it from many languages, and the meaning of Arabic word changes according to the context [4]. The morphological analysis and syntactic complexities (complex word structure and grammar) of Arabic language result in slowing its processing using computerized programs [5]. These limitations i.e. the absence of capital letters or small letters to determine whether the word is a noun or not. Furthermore, the word's place in the sentence is not essential to represent its thematic role. Also, without diacritics (short vowel marks) it is difficult to determine the word position. El-Harby et al. [6] diacritics (vowels) for Qur'an words that lack them, using a bigram Hidden Markov Model (HMM) (HMM, a statistical model for sequence prediction) and a unigram base-line model. According to their research, HMMs are useful instruments for the diacritical restoration work in Arabic. Furthermore, there are a lot of variances in letter formats that might make it challenging to find a letter at the beginning, middle, or end of a word [4].

As a result, the Arabic Text Summarization (ATS) process may involve several stages of text preparation, including normalization, stemming, and morphological analysis, followed by feature extraction, the application of ranking techniques (methods to score sentence importance), and, at the conclusion, the extraction of the summary, based on the complexity of the Arabic language.

Consequently, to summarize a text, we can apply graph-based algorithms to convert a text to a graph. So that the sentences are used as nodes, and the edges are the relationship between those sentences in the graph [7]. This research uses the MST algorithm for extracting text summary to find the most concise set of connected sentences, because it arranges the graph as a tree [7]. The summary process follows several stages beginning with representing the text as a graph, then building the graph, applying MST and finally the text summary could be extracted through selecting MST parent vertex's as a summary. This research is the first to apply three algorithms of MST namely; Prim's (which grows the MST by attaching the nearest vertex) [8], Kruskal's (which sorts edges by weight and adds them without forming a cycle) [9], and Boruvka's (connects components by the smallest outgoing edge iteratively) [10], (three well-known greedy MST algorithms) in single documents extractive ATS. These algorithms are a greedy algorithm (they build the MST by choosing the best local option at each step) that is cheaper, faster, has lower intricacy, and gives the optimal next point in the sorting process, constantly [7]. Each algorithm creates a unique path for MST as it will be presented in Section 4. The dataset used in examining these approaches is EASC (Essex Arabic Summaries Corpus) which is a benchmark dataset of Arabic documents and human-written summaries used to evaluate the performance of Arabic summarization systems.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) which is a set of metrics used to evaluate automatic summaries by comparing them to human-made summaries, particularly focusing on recall, precision, and F-measure.

This paper is structured as follows main reason is highlighted in Section 2, related works is reviewed in Section 3, and the MST algorithm and its use are described in Section 4. An outline of the suggested procedure is given in Section 5. The experiment's findings are examined in Section 6. The research is finally concluded in Section 7.

2. MOTIVATION AND PROBLEM OF STATEMENT

According to the low performance of researches done in graph-based extractive single document ATS, this paper examines the effect of using the MST algorithm for the ATS. MST was not used before in single document extractive ATS and in conformity with the advantages of MST through building a cyclic graph and connects every vertex in the graph with one direct parent. In addition, MST originates fast and gives a suitable solution. Therefore, three algorithms of MST are used to find which one retrieves the best results in ATS.

This research discusses a new approach of graph-based of a document ATS by applying spanning tree with multiple algorithms of MST and shows the performance of the enhanced summary.

3. RELATED WORK

Many studies on text summarization have been carried out throughout the past ten years. In this regard, researches different categorized and classified summarization approaches depending on specific characteristics of every approach.

Researchers such as Hovy and Lin [11] describe a summary as text that corresponds to one or more significant data texts within a document. Also, Mani [12], and others explain summarization of the text as

"a process of getting the key source of information, getting the essential contents and presenting them as a text in template". The Arabic text summarization is done in different approaches which belong to different categories as listed in the rest of this section.

Statistical approaches: in these approaches the summarization is done depending on the values of statistical characteristics such as inverse document frequency and phrase frequency. Douzidia [13], suggested a form of text summarization applying various criteria to get the weight of sentences. A position, combination of frequency, and indicative expression is used to give a conclusion for the sentence [13]. Also, El-Haj [14], who offered four steps are involved in the multi-document extractive text summarization model that extracts Arabic text summary, and they are as follows: obtaining information, Pre-treatment, feature extraction, ranking, scoring, and summary generation. Badry & Moawad [15], is another research which uses query-based research. It starts by creating the weighting matrix that contains the relation between the sentences and the query, then the singular value decomposition on the generated matrix, computations are performed. To choose the top N rated sentences, all of the sentences are then evaluated and ranked. Linguistically or structural approaches: these approaches use linguistic processing methods in the text summarization process Hadni et al. [16] is an example of structural approaches, they proposed a combination of linguistics and statistical approach used as a hybrid approach for Arabic multiword term extraction. Sawalha and Atwell [17], suggested an approach that applies various types of Arabic morphological stemmers and analyzers. The main result that they discovered that for words with three characters, the Khoja stemmer was more accurate than other analyzers. root which is used to compose 80-85% from Arabic words, but the rest of the words are composed up from four, five and six letters [18]. Because the Khoja stemmer has the best accuracy rate for words with three lateral roots and performs well in four letters, it is used as a stemmer in this study. Another approach is the machine learning approaches, which uses machine-learning algorithms. Haboush and Al-Zoubi [19], suggested an extractive summarization of the text algorithm. The researchers applied the root word clustering method as a defining characteristic to summarize the text. El-Sayed and El-Barbary [20], suggested an approach that used a fuzzy linguistic ontology and Field Association words to summarize the Arabic document. Alrahabi et al. [21], suggested a model that depends on linguistic knowledge and identifies, through linguistic indicators, related information as thematic ads, metadata, titles, underlines, etc.

Using graph-based approach, document is represented as a graph using then a graph ranking algorithm is applied, several researches applied graph theory with a method to provide a summary. Mihalcea [3], suggested an extractive graph-based algorithm to rank sentences in the document to be summarized. This new technique obtained competitive results with previously developed state-of-the-art systems. Alami et al. [22], The EASC corpus was used to evaluate a single-document graph-based extractive summary that was proposed using a mixed method to construct a summary of Arabic documents. Yeh et al. [23], suggested a summarization method of graph-based by applying the term of deploying activation th0eory and formulating a comprehensive concept in consideration of network analysis and the essential related vertex's. The algorithm re-weighs sentence essentiality by spreading their sentence-specific characteristic conclusions through the network and hence assessing their essentiality. Al-Taani and Al-Omour [24], suggested an Arabic text summary of extractive graph-based approaches with the shortest path algorithm. The researcher applied many of the key units as stem, word, and n-gram used in summarization. This approach lacks performance; in addition, it does not focus on sentence positions and nouns. Malallah and Ali [25] recommended a method for text summarization that uses modified page rank and linear discriminant analysis (LDA). This approach is designed for publications that are bilingual and contain both Arabic and English. It operates by first classifying sentences into essential and non-important groups using an LDA classifier according to a preset threshold, and then applying the page rank algorithm to the important sentence class. The TAC-2011 dataset was used in the training and testing processes. Al-Abdallah and Al-Taani [26] proposed a single document graph-based approach that use the Firefly algorithm to extract summaries; the EASC is used to measure the summary's performance. Alami et al. [4] try to find the impact of using multiple different stemmers for Arabic text summarization, such as Khoja, Larkey, and Alkhalil's stemmer. Among the Arabic stemmers in use, the researchers discovered that the Khoja stemmer is the best. Elbarougy et al. [27] suggested a summary of an extractive Arabic text approach using the Modified PageRank Algorithm. The researchers represented the document as a graph then making the initial rank for each node is the weight of the edge with the number of nouns is the

sentences' degrees of cosine resemblance. The sentences were arranged based on their final rank after roughly 10,000 iterations of the PageRank algorithm. The summary was then extracted based on the predetermined compression ratio and the redundant sentences are removed from the summary.

Elbarougy et al. [28] discussed the effects of using natural language processing techniques on Performance of the Arabic Summarization Process. This study focuses on the result of removing stop words throughout the preprocessing phase. A graph is used to display the document. and then a summary is extracted. Researchers have found that removing stop words from text can improve summarization performance.

Alselwi and Taşcı [34] addressed the limited research on Arabic extractive text summarization by proposing GEATS, a graph-based method combining Word2Vec embeddings and the PageRank algorithm. Their approach involved pre-processing Arabic texts, extracting semantic features, and constructing a similarity graph to score and rank sentences. Experimental results showed that GEATS outperformed existing methods by 7.5% in F-measure, particularly when using Farasa stemmer and 40% compression. Their work highlights the effectiveness of combining word embeddings with graph-based ranking in Arabic summarization and points to future directions such as incorporating deeper linguistic features and hybrid lemmatization techniques.

Al-Khassawneh and Hanandeh [35] introduced a graph-based extractive summarization method for Arabic texts that evaluates sentences based on relevance, coverage, and diversity. Using statistical and semantic features, their approach showed competitive results on the EASC dataset using ROUGE metrics. This work contributes to Arabic summarization research by integrating multi-dimensional sentence evaluation into graph construction for improved coherence and informativeness.

Eddine et al. [36] proposed AraBART, the first fully pretrained Arabic sequence-to-sequence model for abstractive summarization, based on the BART architecture. AraBART outperformed Arabic BERT-based and multilingual models (mBART, mT5) across several summarization benchmarks, setting a new standard for Arabic abstractive summarization.

Bahloul et al. [37] introduced ArAsummarizer, an unsupervised Arabic text summarization system that segments texts into subtopics and uses an A algorithm on a lexical–semantic graph to select key sentences. Combining statistical, clustering, and graph-based methods, the system achieved strong results on the EASC dataset using various evaluation metrics.

4. MINIMUM SPANNING TREE (MST)

A graph is a mathematical construct that depicts the pairwise relationship between objects. A graph's spanning tree implies a sub-graph that implies all vertices without cycles. This research presents text summarization approach by applying MST algorithms to rank sentences through extracting number of child vertex's in sentences [7]. Generally, spanning trees could be represented in a connected graph, with the minimum possible total edge weight. Figure 1(A) describes a graph and Figure. 1(B) MST of the graph.

Algorithm 1 describes the process of MST [7]. While (G) is the graph that represents the document sentences, (E) number of edges connecting nodes and (M) is the spanning tree vertices group. The mechanism begins by checking whether the tree includes all nodes and the graph results in no cycles. If a sentence has no nodes included in the system, the graph finds its minimum weight, then includes it in the tree. The mechanism to add a new vertex, when found, is to add it to the tree with the already existing vertex. The MST finds the ideal solution with less cost for connecting the graph Vertices without cycles. Here, three types of MST algorithms are compared to find which one returns the best performance of extractive ATS.

Algorithm 1. Basic algorithm for MST [7].

Input: Set of graph vertices (G), set of graph edges (E)
Output: MST (M).
M equals {}
While M does not form a spanning tree:
  find the minimum weighted edge in E that is safe for M
  M equals M union $\{(u,v)\}$
Return (M)

The MST of these different spanning trees depends on the type of the MST algorithm applied.

In this research, three algorithms of MST are used Prim's, Kruskal's and Boruvka's. These algorithms are greedy algorithms as they have a characteristic as a fast MST algorithm.

While Kruskal's algorithm did MST by adding edges to MST, Prim's algorithm builds the MST by adding vertices to the tree. In addition, in Prim's algorithm, which is generally the first vertex to be selected, compared and preferred to but in Kruskal's algorithm, which depends on edges, not vertices. On the other hand, Boruvka starts by forming multiple

Journal of Theoretical and Applied Information Technology
15th July 2025. Vol.103. No.13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MSTs then these MSTs form the major MST [7].

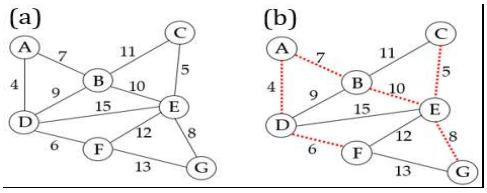

Figure 1. MST basic algorithm example.

4.1. Prim's MST Algorithm

An algorithm that focuses on locating a MST for particular graph, by analyzing all the aspect, whether major or minor, that forms a tree and includes all the nodes within the original paragraph, the algorithm puts into consideration that the importance of edges in the tree is minimum as in formula represented by Formula (1). The algorithm performs by constructing a single vertex of the tree randomly, then, in each step, linking it within the tree to a different vertex that has the slightest possible link.

$$w(M) = \sum_{e \in M} w(e) \quad (1)$$

Where, M: is the resulting MST, which gives a weight $w(e)$ to each edge (e) in the tree. Figure 2 exemplifies the methodology that Prim's algorithm follows which begins in the selected node then grows the tree until it covers all the vertices in the graph. While Figure 2(A) presents the start of Prim's algorithm, Figure 2(F) shows the result of the algorithm, i.e., the MST. Algorithm 2 shows how Prim's algorithm works [8]. The tree starts by an arbitrary node from the graph, after that, its finds the nearest node to the tree in the graph that was not covered in the tree. The process of adding a new node to the tree is repeated until every node in the original graph has been added to the tree., by taking into the account not forming any cycle. (M) represents the tree, (Z) represents nodes of the tree, and (V) represents all nodes in the graph.

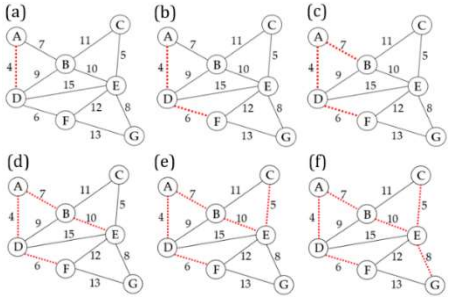

Figure 2. Prim's MST algorithm example.

Algorithm 2. Prim's MST algorithm [8].

Input: A weighted, undirected graph G= *(V, E, w)*.
Output: MST (*M*).
1 *M* equals {};
2 Let *s* be an arbitrarily select vertex from *V*
3 *Z* equals {*s*};
4 **While** |*Z*| strict inequality *n* **do**
5 Find *u* element of *Z and v* element of *V-Z such that the edge (u,v) is smallest edge between Z and V-Z.*
6 *M* equals *M* union {(*u*, *v*)}
7 *Z* equals *Z* union {*v*}

4.2. Kruskal's MST Algorithm

A MST algorithm [9], performs by connecting any two trees in the graph, though they might have the lightest weight, by getting a MST for a linked graph. On the other way, it starts by adding edges to the tree from the least weight every time and continues till all nodes in the tree are linked and by for that no cycles. In Kruskal's algorithm [9], the nodes that exist in the intermediate level of forming the tree might not be fully joined with every other, in the opposite of the Prim's algorithm there isn't add for any edge to the tree if it is not linked directly to it. Algorithm 3 shows how Kruskal spanning tree works [9]. Where (G) is the main graph, (A) edges in the spanning tree, and (V) all vertices in the graph (G). For algorithm 3 the edge is selected the lowest weight in the second iteration the lowest edge from the remaining edge is selected the two edges may not be linked as in Figure 3, and also further edge is selected that doesn't make a cycle, and still doing this action till all vertices are linked to the spanning tree is created. Figure 3 illustrates an example of Kruskal's algorithm starts from Figure 3(A), selects the first edge with the lowest weight, and ends with Figure 3(F) that represents the resulted MST.

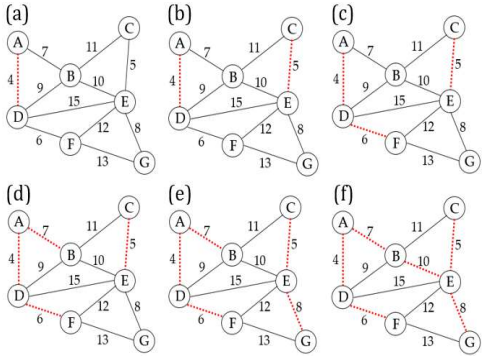

Figure 3. Kruskal's MST algorithm example.

Algorithm 3. Kruskal's MST algorithm [9].

Input: A weighted, undirected graph *G= (V, E, w)*.
Output: MST (*M*).

1	Sort edges in E by weight in a non-decreasing order.
2	*M* **equals** {}
3	Create one set for each vertex.
4	**For each** *edge*(*u,v*) in sorted order **do**
5	*Z* equals find(*u*)
6	*Q* equals find(*v*)
7	**If** *Z* not equal *Q* **then**
8	*M* equals *M* union {(*u,v*)}
9	Union (*Z,Q*)

4.3. Boruvka's MST Algorithm [10]:

An algorithm that performs by locating an MST in a graph with distinct edge weights, or a group of unconnected MSTs.

The Boruvka's algorithm's [10] starts by locating the cheapest edge weight that connects any vertex with another one, assigning its weight edge incident to each graph vertex, then include those in the MST set. Then, the algorithm repeats the process to link the tree to further different tree through getting weight edge for every tree set for a different tree and adding all of those edges to the spanning tree. The algorithm keeps the process until all the sub trees are unified in the spanning tree. Algorithm 4 shows the steps that Boruvka follows [10], where, (*G*) resembles connected directed and weighted graph, MST is the spanning tree. Figure 4 shows a demo example of how Boruvka's algorithm works starting by Figure 4(A) and the final MST is represented by Figure 4(F). as shown in Figure 4(a) the cheapest edge connected with node {A} is edge {AD} and also it is the cheapest edge connected with node{D}, in Figure 4(b) the cheapest edge connected with node {B} is edge {BA}, in Figure 4(c) edge {CE} is the cheapest edge near nodes {C} &{E}, in Figure 4(d) the cheapest edge near node {G} is {GE}, in Figure 4(e) the cheapest edge for node {F} is {FD}, after this step, all the nodes are connected with two spanning trees, so finally we need to find an edge to connect the two trees with each other which is with edge {BE} and there we got the minimum spanning tree.

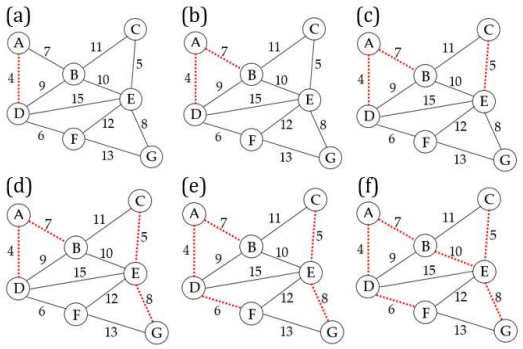

Figure 4. Boruvka's MST algorithm example.

Algorithm 4. Boruvka's MST Algorithm [10].

Input: A weighted, undirected graph G= *(V, E, w)*.
Output: MST (*M*).

1	*M* equals {}
2	**While** |*M*| < *n*-1 **do**
3	*Z* equals a forest consisting of the smallest edge incident to each vertex in *G*
4	*G* equals *G* \ *Z*
5	*M* equals *M* union *Z*

5. The Proposed Approach

This research describes the impact of using three different MST algorithms on different categories of documents in the extracting ATS process.

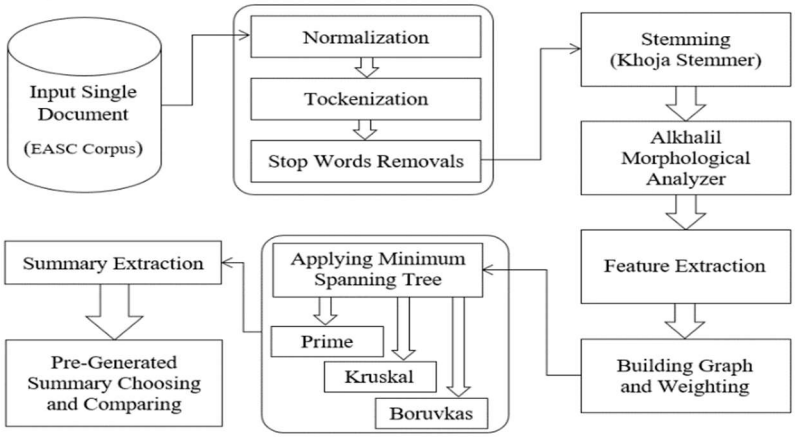

Figure 5. Proposed approach.

Figure 5 shows our suggested approach in full steps of the proposed method that begins with text input, proceeds through preprocessing steps such as tokenization, normalization, and stop word removal, stems morphological analyzers, features extraction, and constructing

then one of the MST algorithms is applied, after that ranking and extracting the summary. These steps are discussed with more details as the followings.

5.1. Pre-processing and Preparation

In this phase, the document is added to the system and examined in the following ways in preparation for the extraction stage:

5.1.1. Normalization

This stage removes numbers and punctuations from the sentence, but leaves the alphabet letters in place. Furthermore, each word's initial (ALEF “ ,أ ,آ !”) is changed to “ا” and each word's final (Taa Marbota “ـة”) is changed to (Heh “ـه”).

5.1.2. Tokenization

The document is separated into paragraphs in this step, which is followed by the division of paragraphs into sentences, and ultimately the division of sentences into tokens, or words, which serve as the fundamental building block for text summary.

5.1.3. Removing of Stop Words

The text is reduced to more helpful terms when they are removed. Additionally, the inefficiency of the weighting procedure is impacted by the non-removal.

5.1.4. Stemming

By using this method, which is described by Khoja and Garside [29], the stemmer ascertains the sentence's word roots and returns each word to its tri-literal root. In order to optimize the computation of terms frequency, this approach minimizes the number of terms in the document.

5.1.5. Morphological Analysis

Because of its accuracy, which surpassed 98%, the Safar (AL Khalil) morphological analyzer [30] is employed as a morphological analyzer. In morphological analysis, there are many tasks to do like stemming, stop words removals, and part of speech tagging. Stemming and stop words removals were done in previous steps from this research, but the part of speech tagging is not, so in this step the part of speech tagging is the main issue here. So, the morphological analyzer gives every word a tag that represents its Part-of-Speech (POS) position in the sentence [29] like noun, verb, article and so on. So the main task of morphological analyzer is a part of speech tagging. The Sentences which have many numbers of nouns take values higher than the others. Table 1 provides an example of part-of-speech tagging and morphological analysis. The phrase has three nouns, one adjective, two verbs, and two stop words, according to Table 1. Every sentence in the document morphological analysis is subjected to this examination.

TABLE 1. PART OF SPEECH TAGGING EXAMPLE

و تم بناء السد العالي على نهر النيل							
و	تم	بناء	السد	العالي	على	نهر	النيل
Stop word	Verb	Verb	Noun	Adjective	Stop word	Noun	Noun

5.2. Features Extraction

The traits that are used in the weighting process of the graph in this stage are extracted. This step makes the term identical to the root word. Here are the three functions:

5.2.1. Sentence Position

According to Hovy and Lin [11], sentence order in the text is a reflection of the importance of the sentence. As for this characteristic, every sentence from the sentences in the introduction and conclusion will possess an extra weight than other sentences.

5.2.2. Cosine Similarity Between Two Sentences

This characteristic followed stemming and stop words removal steps to locate the similarity through getting (TF-IDF) [31] and the mutual words between two sentences. Formula (2) provides an example of how to calculate (Term Frequency). The calculation of (Inverse Document Frequency) is shown in formula (3). The computation of (TF-IDF) for the term “*t*” is shown in formula (4). The computation of (TF-IDF) for a sentence, which is the sum of (TF-IDF) for each word in the sentence, is shown in formula (5). The cosine similarity between sentence

1 (*Si*) and sentence 2 (*Sj*) in the document can be calculated using formula (6), where "*m*" is the number of mutual words between the two sentences and "k" is the word offset in the mutual list; the term number "*k*" in the mutual list in (*Si*) is represented by TF-IDF(t_{jk}), and the term number "k" in the mutual list in (*Sj*) is represented by TF-IDF(t_{jk}). In other words, to find the cosine similarity between sentences (*Si*) and (*Sj*), we first use Formula (4) to calculate (TF-IDF) for each term in both phrases. (2) Locate the list of terms that are used in both sentences; this list has a length of "*m*" .(3) using Formula (5) while iterating through the mutual list. (2).

$$\text{TF}(t) = \frac{\text{Number of occurrences of term } t \text{ in document}}{\text{Total number of all terms in the document}} \quad (2)$$

$$\text{IDF}(t) = \log\left(\frac{\text{Number of all sentences in the document}}{\text{Number of sentences containing the term } t}\right) \quad (3)$$

$$\text{TF-IDF}(t)=\text{TF}(t)*\text{IDF}(t) \quad (4)$$

$$\text{TF-IDF}(s)=\sum_{t\in s}\text{TF-IDF}(t) \quad (5)$$

$$\text{Cosine\_Similarity}(Si,Sj) = \frac{\sum_{k=1}^{m}\text{TF-IDF}(t_{ik})*\text{TF-IDF}(t_{jk})}{\sqrt{\sum_{k=1}^{m}\text{TF-IDF}(t_{ik})^2}*\sqrt{\sum_{k=1}^{m}\text{TF-IDF}(t_{jk})^2}} \quad (6)$$

1) The Count of Nouns in Each Sentence

That derives from the morphological analysis step. The calculation process of the nouns measure can be shown, Formula (7), by counting the mutual nouns in two phrases and dividing the result by the total number of nouns in the two sentences.

$$\text{Nouns\_Measure}(Si,Sj) = \frac{|\text{Nouns\_List}(Si)\cap\text{Nouns\_List}(Sj)|}{|\text{Nouns\_List}(Si)\cup\text{Nouns\_List}(Sj)|} \quad (7)$$

Formula (8) shows the calculation of the similarity between two sentences, which depends on Formula (6) and (7) to calculate cosine similarity and nouns measure respectively. The result of Formula (8) will be used as the edge weight which connects between these two sentences in the graph.

$$\text{Similarity}(Si,Sj) = \text{Cosine\_Similarity}(Si,Sj) + \text{Nouns\_Measure}(Si,Sj) \quad (8)$$

5.3. Extract and Build the Summary

Through this stage, you can build the graph model then the weighting and extracting the texts. Finally, extracting and building the summary. This stage is done in consequence with the next step:

5.3.1. Building Graph and Weighting

Through this process, you can build the graph where sentences represent the vertices and a connecting edge exists between any two found nodes. The weight of the edge is calculated as in Formula (8).

5.3.2. Apply MST

Through this step, MST is used by applying three algorithms separately, namely Prim's, Kruskal's and Boruvka's algorithm. The graph is represented as a tree beginning with the node named root and the first sentence of the document is added into that node, as illustrated in the next parts. The current research uses the following Algorithm 5. As the algorithm describes, the input document is fed to the system then it describes the summarization by doing NLP then extracting the needed characteristics and constructing the graph, after generating the graph the MST is applied then lastly the summary is extracted.

Algorithm 5. Proposed approach for text summarization.

Input: Single document.
Output: The summary.
Collection the Max Sentences in the Summary ← Total Sentences in Document.
Document_Category ← Detect Document Category (*health, education, … etc.*)
Morphological analyzer (Alkhalil)
Graph ← New Graph ()
Foreach Sentence: Document.Sentences
Normalization ()
Tokenization ()
StopWordsRemoval ()
Stemming ()
S_TF-IDF←Calculate Sentence TF-IDF ()
S_Noun_List ← Applying Morphological Analyzer & Get Nouns List ()
New_Node ← CreateGraphNode (S_TF-IDF, S_Noun_List)
Graph.add (New_Node)

```
14  Foreach Node: Graph.Nodes
15  If (Node <> New_Node)
16  Similarity ← Cosine_Similarity (Node, New_Node)
17  Nouns_Measure ← Noun_Calc (Node, New_Node)
    Graph.CreateEdge(Node, New_Node, Similarity, Nouns_Measure)
18    Apply_MST (Prim)
19    Apply_MST (Kruskal)
20    Apply_MST (Boruvka)
21    summary ←Extract_summary(Compression_Ratio)
22    summary←Removing_reduandancy(summary)
23    Return summary
```

5.3.3. Summary Extraction

After applying the MST algorithm, nodes could be sorted in conformity to the other nodes connected to it, weight is to be added to the sentence for its original text arrangement. After that, the sentences are sorted again and the summary could be extracted depending on the compression ratio.

5.3.4. Removing Redundancy

Repetitive sentences should be eliminated throughout the summary extraction process. A sentence is considered redundant if there is a significant amount of overlap more than 90% between it and any other sentence in the summary.

5.3.5. Choose the Pre-generated Summary Files and Compare with it

Through this step, the pre-generated summaries are compared with resulted summary.

Figure 6 highlights the flowchart of suggested approach that begins with inputting the suggested document, then normalizing the text through removing punctuations, digits and characters. In the stage of removing stop words and stemming happens. Furthermore, morphological processing happens by applying Safar (AlKhalil) morphological analyzer. After that the document is displayed as a graph $G(V,E)$ with G: graph, V: a set of graph nodes represent the document sentences, E: represents edge that's link among graph nodes. Through the next stage, MST by its three algorithms is applied and a summary is generated individually. In the end, pre-generated summaries are selected; the measurement metrics are calculated and stored for comparison processes.

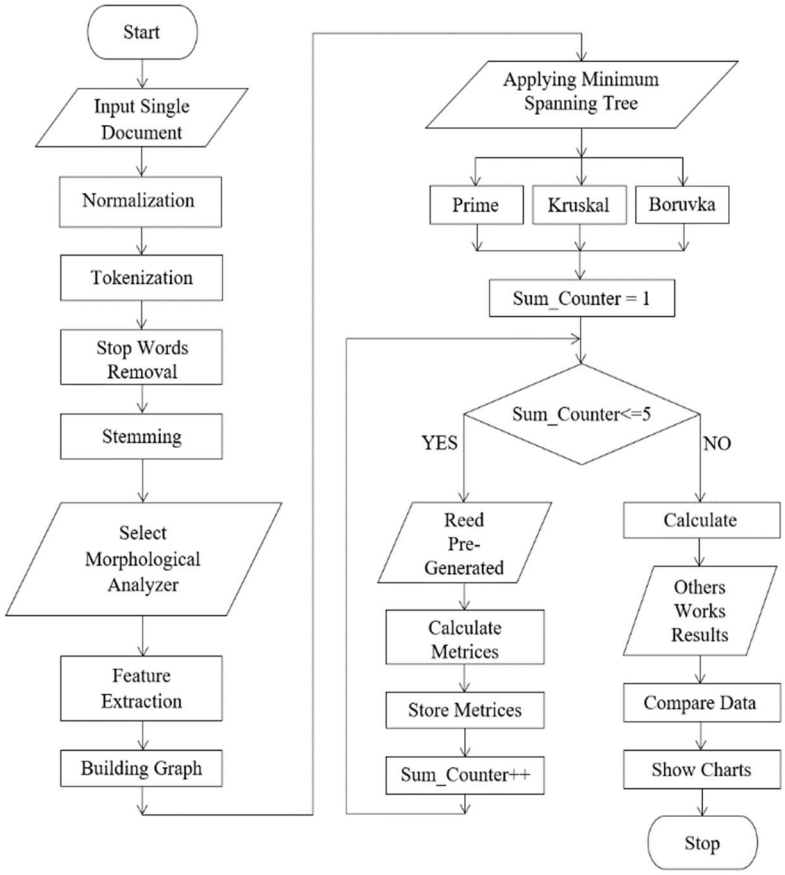

Figure 6. Proposed approach flow chart.

6. EXPERIMENTATION AND RESULTS

6.1. Dataset (Corpus)

The approach evaluated, the Essex Arabic Summaries Corpus (EASC) [32,33], which is employed as a standard corpus. There are (153) texts in the corpus, with five summaries for each document, and (765) Arabic human-made summaries [32]. EASC covers ten subjects: environment, politics, art and music, health, finance, science, sports and technology, religion, education and, tourism. Three summaries for each document are extracted by the system in conformity to the selected MST algorithm.

6.2. Evaluation Metrics

Three key performance indicators precision, recall, and F-measure are used to calculate the summary performance. This comparison is done in accordance with Formulas (9), (10) and (11), respectively.

- *Precision*: To gauge the text's size that the system has returned.

$$\text{Precision}=\frac{\text{Extracted Summary} \cap \text{Provided Summary}}{\text{Extracted Summary}} \quad (9)$$

- *Recall*: The metric coverage system reflects the ratio of the extracted relevant sentences.

$$\text{Recall}=\frac{\text{Extracted Summary} \cap \text{Provided Summary}}{\text{Provided Summary}} \quad (10)$$

- *F-measure*: Works a balance relation among recall metric and precision metric.

$$\text{F-measure}=\frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}} \quad (11)$$

6.3. Experiment Setup

Through this subsection, the summary process of a text from the EASC corpus is shown in the next lines and subsections. Three different algorithms of

MST, using a different way to build the minimum spanning tree, are used here to find the best. Figure 7 highlights an instance of an Arabic single document that contains nine sentences. Then in Figure 8, this document can be changed as a fully joined graph.

S01	لوتشانو بافاروتي مواليد مودينا في 12 أكتوبر 1935 -توفي بمودينا في 6 سبتمبر 2007، مغني تينور إيطالي، يعد من أشهر فنانين الأوبرا في الطبقة العالية من الرجال في عصرنا الحاضر، وأحد التينور الثلاثة وقد ولد في مدينة مودينا.
S02	ولد في بمدينة مودينا شمال إيطاليا لعائلة خباز، بعد أن تخلى عن حلمه بأن يصبح حارس مرمى كرة قدم محترف، ثم قضاء سبعة سنوات في التدريب الصوتي، بدأ بافاروتي حياته كتينور في إيطاليا عام 1961، ثم بدأ الغناء في دور الأوبرا فيهولندا و فيينا ولندن وأنقرة وبودابست وبرشلونة.
S03	وقد اكتسب التينور الشاب حينها الخبرة القيمة بالإضافة إلى التقدير الواضح.
S04	وفي أثناء عروضه للولايات المتحدة على دعوة من السوبر انوساتراند عام 1965 ذاع صيته وثبت أقدامه على الساحة العالمية بين عامي 1966 و1972 حيث أدى في أكبر دور الأوبرا في العالم مثل La Scala في ميلان.
S05	بحلول منتصف السبعينيات، صار بافاروتي مشهورا في جميع أنحاء العالم بتميز وروعة صوته خاصة الطبقات العليا.
S06	وبين عامي 1970 و1980 كان بافاروتي قد اثبت نفسه كأحد أعظم مطربي التينور بعروضه المتعددة في أكبر دور الأوبرا في العالم.
S07	كما ذاع صيته خارج نطاق جمهور الأوبرا كنجم غنائي في عام كأس العالم بإيطاليا خاصة بعد أدائه الرائعة لأريا Nessun Dormaمن أوبرا Turandot كأحد التينور الثلاثة في حفلهم الأول عشية نهائي كأس العالم، غنى فيها بافاروتي مع نجمي التينور بلاسيدو دومينجو وخوسيه كاريراس ومعا حققوا شهرة طاغية ونجاحا عظيما في جميع أنحاء العالم.
S08	السنوات التالية لاقت تقلصا في عدد أدائه للأوبرات على المسرح بسبب الزيادة المفرطة في وزنه وكان آخر أداء في أوبرا له في مارس عام 2004 في Mets.
S09	وفي الأوليمبيات الشتوية في تورينو Turin عام 2006 شاهدته إيطاليا والعالم يغني للمرة الأخيرة حيث غنى بافاروتي Nessun Dorma وأدت الجموع الكبيرة دور الكورال المصاحب، في أداء مهيب مؤثر.

Figure 7. Example of Arabic single document.

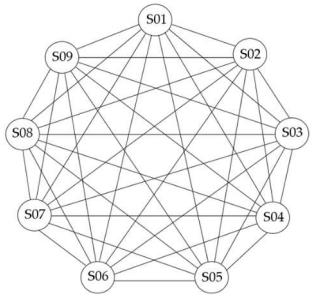

Figure 8. Document as a graph.

A closer look at Figure 7 shows the following:

- The final summary extracted when using Prim's algorithm of MST resulted in the following sentences “S01, S02 and S04”.
- The final summary extracted when using Kruskal's algorithm of MST resulted in the following sentences “S01, S02 and S09”.
- The final summary extracted when using Boruvka's algorithm of MST resulted in the following sentences "S01, S02 and S03".

6.3.1. MST with Category and Morphological

Table 2 illustrates the results of applying the suggested approach in conformity with the document category and the three MST algorithms. Depending on average results, Kruskal’s MST algorithm gets the best F-measure among the others in all categories except health and environment categories in which Boruvka's MST algorithm had the best F-measure. As described in bold numbers. To sum up, Boruvka's algorithm gets the best results, while Kruskal's algorithm precision results were better than the two other algorithms.

6.3.2. Average Results According to MST Algorithms only

This subsection is about applying multiple MST results ratio while neglecting the kind of document category. Table 3 shows the final results of the system by getting the precision, recall, and F-measure for every MST algorithm. Based on the final results Kruskal's algorithm showed the best results because it depends on edges weight in the process of building spanning tree, so it returns the results depending on adding edges to the tree which is the relation between sentences as mentioned in Kruskal’s MST algorithm subsection. So Kruskal’s results are used in the comparison with other results. In addition, depending on the results, Prim's algorithm results are the lowest. Furthermore, the results of Kruskal is the based on the weight of the edges in the building spanning-tree process, so it returns the results depending on the relation among sentences. Figure 9 illustrated the ratio of results of all metrics for the three MST algorithms.

TABLE 2. EVALUATION METRICS RESULTS.

Document Category	MST Type	Precision	Recall	F-Measure
Science and Technology	Kruskal	72.75	76.84	70.91
	Boruvka	67.02	79.97	68.63
	Prim	66.92	42.29	46.27
Education	Kruskal	71.93	95.23	78.71
	Boruvka	55.72	96.42	68.90
	Prim	67.26	79.76	71.70
Health	Kruskal	71.23	75.32	68.99
	Boruvka	65.67	86.47	71.81
	Prim	70.20	53.06	54.63
Sport	Kruskal	69.82	83.33	72.42
	Boruvka	62.59	84.76	68.59
	Prim	65.83	53.12	53.14
Art and Music	Kruskal	65.11	79.17	68.41
	Boruvka	59.30	85.33	65.53
	Prim	67.11	57.67	55.43
Environment	Kruskal	57.73	62.69	55.52
	Boruvka	55.94	67.52	56.59
	Prim	47.47	33.13	34.31
Finance	Kruskal	84.28	91.58	86.07
	Boruvka	71.34	95.02	79.69
	Prim	79.60	78.86	74.92
Politics	Kruskal	75.00	84.89	77.14
	Boruvka	63.75	88.58	71.65
	Prim	80.35	59.63	64.27
Religion	Kruskal	58.47	93.75	70.66
	Boruvka	55.91	95.00	69.26
	Prim	42.71	35.53	36.18
Tourisms	Kruskal	54.54	87.24	65.37
	Boruvka	49.84	81.73	60.51
	Prim	70.36	50.36	56.09

TABLE 3. FINAL EVALUATION RESULTS.

MST Algorithm	Precision	Recall	F-Measure
Kruskal	68.09	83	71.42
Boruvka	60.71	86.08	68.12
Prim	65.78	54.34	54.69

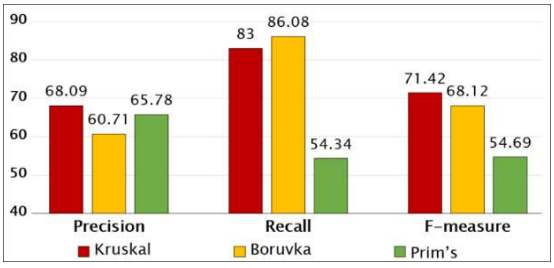

Figure 9. Final evaluation results of Precision, Recall and F-measure for the three MST algorithms.

Three issues were taken into account when selecting the comparison papers: (1) the selected paper should use the same data set in the evaluation process. (2) the selected paper should use the same evaluation metrics. (3) the selected paper should be a state-of-art paper or at least using a graph-based algorithm. Table 4 compares the current study results with other four studies entitled (1) using semantic and analysis for ATS [22], research (2) investigating various stemmer types on Arabic text [4], (3) employing a graph-based method to Arabic text summarization with shortest path algorithm [24], and (4) utilizing firefly algorithm [26].

All the previous researches as well as the current research are using the EASC corpus as a dataset.

This research results ratio shows better performance in all the metrics compared to the other

studies as discussed in Table 4.

TABLE 4. COMPARISON WITH OTHER RESEARCH.

Methods	Precision	Recall	F-measure
Statistical and Semantic Analysis [22]	57.62	58.80	58.20
Different Types of Stemmers [4]	55	48	51
Shortest Path Algorithm [24]	54	47	51
FireFly Algorithm [26]	57.32	60.14	57.52
The Proposed Method (Kruskal's Algorithm)	68.09	83	71.42

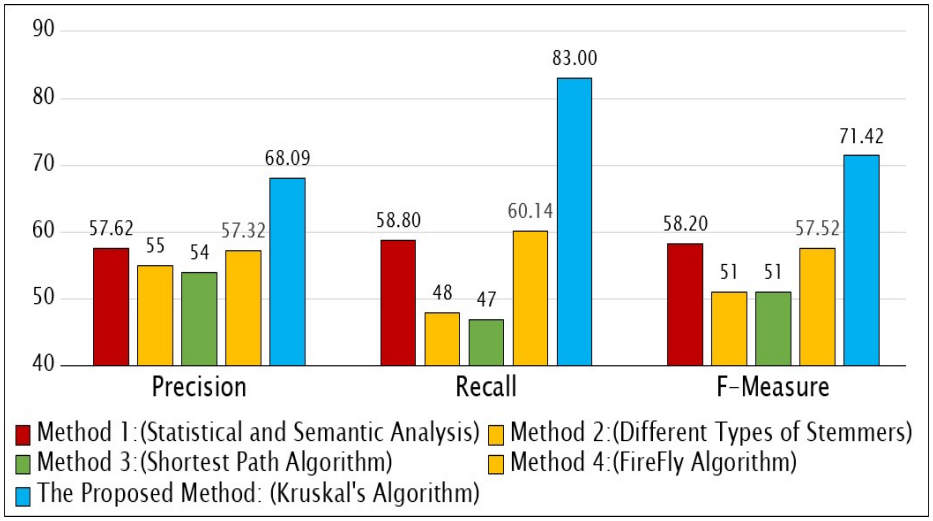

Figure 10. Performance evaluation compared with other research.

Figure. 10 introduces this research's performance evaluation of results compared with other Researches. The Safar morphological analysis proved to enhance performance in all metrics compared to other researches. In the first metric "Precision" this research percentage was 68%. In the second metric "Recall" this research performance was 83%. In the third metric "F-measure", this research performed 71.4%.

The enhancement of the performance in the current research can be reverted to the fact that MST performs accurately because it is fast and without cycles as mentioned before. Finally, the results show that Kruskal's algorithm has the best results compared with the other three algorithms in all document categories, except health and environment. Prim's algorithm gives the lowest results among all the algorithms that are applied. Generally, the average shows that Kruskal gave the best results because it depends on edges' weight in the process of building a spanning tree, and returns the results according to the relationship between the sentences.

7. CONCLUSIONS

Text Summarization takes its significance from the significance of the internet, the web and electronic libraries. The graph-based method is used in the text summarization process. There are many methods that can be applied to extract the final summary for the graph-based approaches. This paper aims to improve the resulting summaries performance by putting forward three distinct MST algorithms for the ATS rooting process. By reading the content, normalizing the data, eliminating stop words, stemming, using a morphological analyzer, and finally applying the graph to obtain the summary, the summarization thresholds process the summary extraction process depends on two factors contraction rate and removing redundancy

The metrics used here are Precision, Recall, and F-measure. In general, when Kruskal's algorithm is used, it gives results better than the others. The results are compared with three other inquiries using the same dataset (EASC corpus).

The results of Kruskal's algorithm are the best because it depends on edges weight in the process of

building a spanning tree, so it returns the results depending on the relation between sentences.

8. REFERENCES

[1] E. Lloret, "Text Summarization based on Human Language Technologies and its Applications," *Journal of Natural Language Processing*, vol. 48, pp. 119-122, 2011.

[2] E. Lloret, and M. Palomar, "Text summarization in progress: a literature review," *Artificial Intelligence Review*, vol. 37, no 1, pp. 1-41, 2012.

[3] R. Mihalcea, "Graph-based ranking algorithms for sentence extraction, applied to text summarization," In Proceedings of the ACL 2004 on Interactive poster and demonstration sessions. Association for Computational Linguistics, page 20, 2004.

[4] N. Alami, M. Meknassi, S.A. Ouatik, and N. Ennahnahi, "Impact of stemming on Arabic text summarization," In Information Science and Technology (CiSt), 2016 4th IEEE International Colloquium on, pp. 338-343, 2016.

[5] K. Ganapathiraju, J. Carbonell, and Y. Yang, "Relevance of Cluster size in MMR based Summarizer," A Report 11-742 Self-paced lab in Information Retrieval, 2002.

[6] A. A. El-Harby, M. A. El-Shehawey, R. A. El-Barogy, "A statistical approach for Qur'an vowel restoration," *ICGST-AIML Journal*, vol. 8, no. 3, PP 9-16, 2008.

[7] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms. MIT press.

[8] R. C. Prim, "Shortest connection networks and some generalizations," *Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[9] J.B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48-50, 1956.

[10] J. Nešetřil, E. Milková, and H. Nešetřilová, Otakar Borůvka "on minimum spanning tree problem translation of both the 1926 papers, comments, history," *Discrete Mathematics*, vol. 233, no.1-3, pp. 3-36, 2001.

[11] E. Hovy, and C. Y. Lin, "Automated text summarization in SUMMARIST," MIT Press, pp. 81-94, 1999.

[12] I. Mani, "Automatic Summarization," John Benjamin's Publishing Co. pp. 1-22, 2001.

[13] F.S. Douzidia, "Automatic summarization of Arabic text," Memory presented at the Faculty of Graduate Studies, University of Montreal, 2004.

[14] M. El-Haj, "Multi-document Arabic Text Summarization," Ph.D. Thesis, University of Essex, 2012.

[15] R. M. Badry, and I. F. Moawad, "A Semantic Text Summarization Model for Arabic Topic-Oriented," In International Conference on Advanced Machine Learning Technologies and Applications, Springer, Cham, pp. 518-528, 2019.

[16] M. Hadni, A. Lachkar, and S. Ala, "Multi-Word Term Extraction based on New Hybrid Approach for Arabic Language," Computer Science & Information Technology (CS & IT), pp. 109-120, 2014.

[17] M. Sawalha, and E. Atwell, "Comparative evaluation of Arabic language morphological analysers and stemmers," Colling 2008: Companion volume: Posters, pp. 107-110, 2008.

[18] S. Eldin, "Development of a computer-based Arabic lexicon," In proceedings of the Int. Symposium on Computers & Arabic Language (ISCAL) Riyadh, KSA, 2007.

[19] A. Haboush, and M. Al-Zoubi, "ATS Model Using Clustering Techniques," *World of Computer Science and Information Technology Journal (WCSIT)*, vol. 2, no. 3, pp. 62-67, 2012.

[20] A. El-Sayed, and O. El-Barbary, "Arabic document summarization using FA fuzzy ontology," *International Journal of Innovative Computing, Information and Control*, vol. 10, no. 4, pp. 1351-1367, 2014.

[21] M. Alrahabi, G. Mourad, and B. Djioua, "Semantic filtering of texts in Arabic for a prototype of automatic summarization, the automatic processing of Arabic," JEP-TALN 2004, Fès, 2004.

[22] N. Alami, Y. El Adlouni, N. En-nahnahi, and M. Meknassi, "Using statistical and semantic analysis for Arabic text summarization," In International Conference on Information Technology and Communication Systems," Springer, Cham, pp. 35-50, 2017.

[23] J. Y. Yeh, H. R. Ke, and W. P. Yang, "iSpreadRank: Ranking sentences for extraction-based summarization using feature weight propagation in the sentence similarity network," *Expert Systems with Applications*, vol. 35, no. 3, pp. 1451-1462, 2008.

[24] A. T. Al-Taani, and M. M. Al-Omour, "An extractive graph-based Arabic text summarization approach," In the International Arab Conference on Information Technology, Jordan, 2014.

[25] S. Malallah, and Z. H. Ali, "Multilingual Text Summarization based on LDA and Modified PageRank," *Iraqi Journal of Information Technology*, vol. 9, no. 3, pp.139-160, 2019.

[26] R. Z. Al-Abdallah, and A. T. Al-Taani, "Arabic Text Summarization using Firefly Algorithm," In 2019 Amity International Conference on Artificial Intelligence (AICAI), IEEE, pp. 61-65, 2019.

[27] R. Elbarougy, G. Behery, & A. El Khatib, "Extractive Arabic Text Summarization Using Modified PageRank Algorithm," *Egyptian Informatics Journal*, doi.org/10.1016/j.eij.2019.11.001, in press.

[28] R. Elbarougy, G. Behery, & A. El Khatib, "A Proposed Natural Language Processing Preprocessing Procedures for Enhancing Arabic Text Summarization," In Recent Advances in NLP: The Case of Arabic Language, Springer, Cham, vol. 874, pp. 39-57, 2019.

[29] S. Khoja, and R. Garside, "Stemming Arabic Text," Computing Department, Lancaster University, Lancaster, U.K., 1999.

[30] Y. Jafar, and K. Bouzoubaa, "Benchmark of Arabic morphological analyzers challenges and solutions," Intelligent Systems: Theories and Applications (SITA-14), 2014 9th International Conference on. IEEE, 2014.

[31] A. Abu-Errub, "Arabic Text Classification Algorithm using TF.IDF and Chi Square Measurements," *International Journal of Computer Applications*, vol. 93, no. 6, pp. 40-45, 2014.

[32] M. El-Haj, U. Kruschwitz, and C. Fox, "Using Mechanical Turk to create a corpus of Arabic summaries," 2010.

[33] A. Elnahas, M. Nour, N. El-Fishawy, and M. Tolba, "Machine Learning and Feature Selection Approaches for Categorizing Arabic Text: Analysis, Comparison, and Proposal", *Egyptian Journal of Language Engineering,* Vol. 7, No. 2, 1-19,2020

[34] Alselwi, G., & Taşcı, T. (2024). Extractive Arabic Text Summarization Using PageRank and Word Embedding. Arabian Journal for Science and Engineering, 49(9), 13115-13130.

[35] AL-Khassawneh, Y. A., & Hanandeh, E. S. (2023). Extractive Arabic text summarization-graph-based approach. Electronics, 12(2), 437.

[36] Eddine, M. K., Tomeh, N., Habash, N., Roux, J. L., & Vazirgiannis, M. (2022). Arabart: a pretrained arabic sequence-to-sequence model for abstractive summarization. arXiv preprint arXiv:2203.10945.

[37] Bahloul, B., Aliane, H., & Benmohammed, M. (2020). ArA*summarizer: An Arabic text summarization system based on subtopic segmentation and using an A* algorithm for reduction. Expert Systems, 37(2), e12476.