Birzeit University
Faculty of Engineering & Technology
Department of Electrical & Computer Engineering
ENCS5300, 37

Enhancing Graph-Based Arabic Extractive Text Summarization using Semantic and Statistical Features

Prepared By:
Lama Naser 1200190
Maha Mali 1200746
Anas Naji 1200231

Supervised By:
Dr. Aziz Qaroush

A Graduation Project submitted to the Department of Electrical and Computer Engineering in partial fulfilment of the requirements for the degree of B.Sc. in Computer Engineering

Birzeit
July 2025

Abstract

In today's digital world, managing large amounts of textual data is a growing challenge, especially with the rise of information overload. Automatic Text Summarization (ATS) offers a solution by condensing information into concise summaries while preserving the key points. This report focuses on Arabic text summarization, proposing a graph-based extractive method tailored to handle the unique characteristics of the Arabic language. The method combines semantic and structural features to improve the quality and relevance of generated summaries.

Our approach employs advanced techniques for sentence representation and ranking. For sentence representation, we use FastText embeddings and AraBERT, a deep learning model designed for the Arabic language. These methods capture semantic and contextual information more effectively than traditional techniques. For ranking, we introduce modified algorithms, such as Weighted Degree Centrality and Semantic TextRank (ST-Rank), which leverage both the structure of the text graph and the semantic relationships between sentences.

The Essex Arabic Summaries Corpus (EASC) is chosen as the dataset for experiments due to its diversity and the availability of human-written reference summaries. To evaluate the quality of the generated summaries, we use the ROUGE metric, which measures content overlap with reference summaries. Our proposed method achieved ROUGE-1 F-measure score of **0.7031**, surpassing existing benchmark techniques. Also, our model converged approximately **5.15**$\times$ faster than PageRank and **4.85**$\times$ faster than TextRank, highlighting its advantage in computational efficiency. This trade-off between output quality and computation speed is especially valuable for large-scale or real-time applications.

المستخلص

في عالمنا الرقمي اليوم، أصبح التعامل مع كميات كبيرة من النصوص تحديًا متزايدًا، خاصة مع تزايد تدفق المعلومات. توفر تقنية التلخيص التلقائي للنصوص (Automatic Text Summarization - ATS) حلاً من خلال تلخيص المعلومات بشكل مختصر مع الحفاظ على النقاط الرئيسية. يركز هذا التقرير على تلخيص النصوص العربية ويقترح منهجية تعتمد على التلخيص الاستخراجي باستخدام الرسوم البيانية، مصممة للتعامل مع الخصائص الفريدة للغة العربية. يجمع المنهج بين الميزات الدلالية والبنيوية لتحسين جودة الملخصات المنتجة.

تستخدم طريقتنا تقنيات متقدمة لتمثيل الجمل وترتيبها. بالنسبة لتمثيل الجمل، نستعين بـ **FastText embeddings** ونموذج **AraBERT**، وهو نموذج تعلم عميق مصمم خصيصًا للغة العربية. تساعد هذه التقنيات في التقاط المعلومات السياقية والدلالية بشكل أكثر فعالية مقارنة بالطرق التقليدية. بالنسبة للترتيب، نقدم خوارزميات معدلة مثل **Weighted Degree Centrality** و **Semantic TextRank (ST-Rank)** التي تعتمد على بنية الرسم البياني النصي والعلاقات الدلالية بين الجمل.

تم اختيار مجموعة بيانات **Essex Arabic Summaries Corpus (EASC)** لإجراء التجارب نظرًا لتنوعها وتوفر ملخصات مرجعية مكتوبة يدويًا. تم تقييم جودة الملخصات الناتجة باستخدام مقياس **ROUGE**، الذي يقيس مدى التداخل بين الملخصات المنتجة والملخصات المرجعية. حقق المنهج المقترح أعلى نتيجة في مقياس **ROUGE-1 F-measure** بلغت **0.7031**، متفوقًا بذلك على العديد من التقنيات السابقة. كما أظهر النظام كفاءة حسابية ملحوظة، حيث كانت سرعة التقارب لديه أعلى بحوالي **5.15 مرة** من خوارزمية PageRank و**4.85 مرة** من TextRank، مما يبرز كفاءته العالية في الأداء. وتُعد هذه الموازنة بين جودة النتائج وسرعة المعالجة ذات أهمية كبيرة في التطبيقات الواسعة أو التي تتطلب تنفيذًا في الوقت الحقيقي.

Contents

List of Tables

List of Figures

Acknowledgement

The authors acknowledge the use of OpenAI ChatGPT for grammatical revision and improve the language quality of this report. After leveraging this tool, the authors carefully reviewed and edited the content to ensure its accuracy, alignment with the research objectives, and compliance with academic standards. The authors take full responsibility for the final content of this publication.

List of Abbreviations

ATS Automatic Text Summarization

NLP Natural Language Processing

LSA Latent Semantic Analysis

SRL Semantic Role Labeling

LDA Latent Dirichlet Allocation

GA Genetic Algorithm

PSO Particle Swarm Optimization

ACO Ant Colony Optimization

TF-ISF Term Frequency-Inverse Sentence Frequency

TF-IDF Term Frequency-Inverse Document Frequency

ROUGE Recall-Oriented Understudy for Gisting Evaluation

KPS Key Phrase Score

KPL Key Phrase Length

ST-Rank Semantic TextRank

EASC Essex Arabic Summaries Corpus

Chapter 1

Introduction and Motivation

Contents

1.1 Motivation

In today's digital age, the huge volume of information generated daily has reached unprecedented levels, particularly in domains such as academic research, news, social media, and customer support. Over 2 million academic articles are published annually in more than 46,000 journals worldwide, with the number increasing at an average yearly rate of 5% [1]. This enormous growth makes it increasingly challenging for scholars to stay updated with the ever-expanding body of knowledge [2]. Automatic Text Summarization (ATS) has become essential in addressing this challenge, offering tools to compress long and complex studies into concise, readable summaries. By quickly extracting the main ideas. ATS helps users manage large amounts of information efficiently, saving time and enabling them to focus on the most relevant content.

The need for concise and accessible information extends beyond academic research to other areas of daily life. Nearly 93% of internet users access news online, with many preferring shorter, more digestible formats [3]. For instance, two-thirds (66%) of global internet users watch short news videos weekly, with this figure rising to 87% in regions like Thailand [3]. ATS meets these demands by using algorithms to condense texts while preserving key points, reducing less relevant details [4]. This technology is widely applied in fields such as news aggregation, customer service, academic research, and the legal and medical sectors. By providing brief, meaningful summaries, ATS enhances productivity, supports faster decision-making, and makes information more accessible for users overwhelmed by information overload.

Automatic text summarization can be classified based on various aspects, as illustrated in Figure 1.1, with a common classification focusing on the type of output: extractive, abstractive, and

hybrid. Extractive summarization selects key sentences directly from the original text to create a summary, offering simplicity, speed, and accuracy by retaining the exact wording of the source. However, it may result in redundancy or lack smooth transitions between sentences. Abstractive summarization, on the other hand, generates new sentences by understanding the text's meaning and rewriting it in a concise and clear manner, much like human summarization. While this approach produces more natural and readable summaries, it is computationally demanding, more complex, and relies on advanced techniques like deep learning. Hybrid summarization combines the strengths of both methods by initially using extractive techniques to identify key sentences and subsequently applying abstractive methods to refine or rephrase them, aiming to provide accurate, fluent, and coherent summaries by combining the strengths of both methods [5].

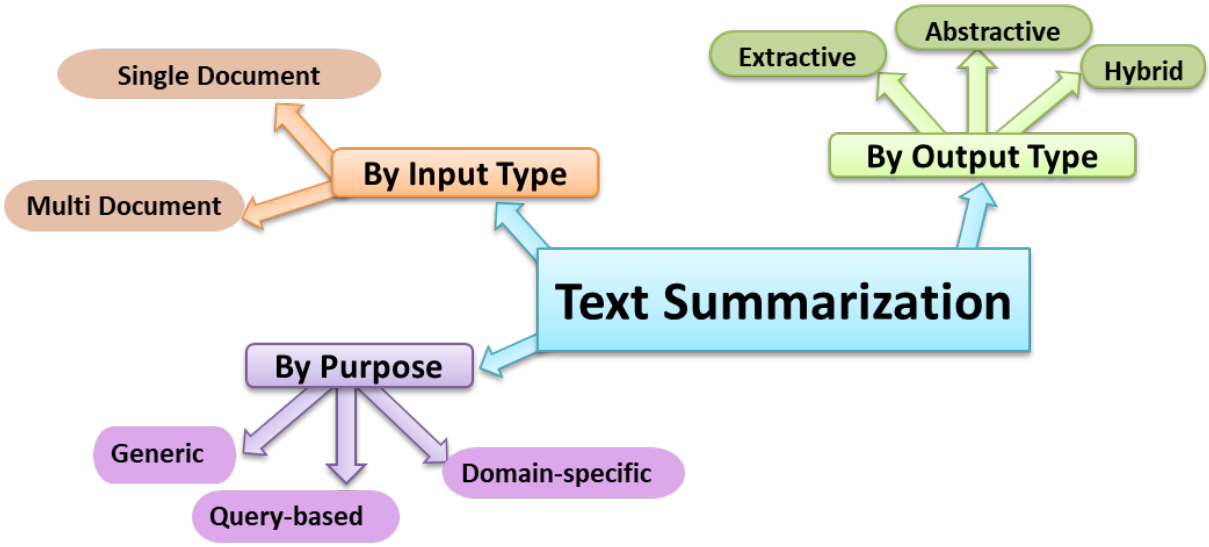

Figure 1.1: Text summarization taxonomy

Text summarization can also be categorized based on the type of input, namely single-document summarization and multi-document summarization. Single-document summarization involves generating a summary from a single text source, making it suitable for summarizing individual research papers, articles, or news stories. In contrast, multi-document summarization integrates information from multiple texts to produce a comprehensive summary that encompasses diverse perspectives. This approach is particularly valuable for tasks such as policy analysis or competitive intelligence, where synthesizing various viewpoints is essential [6].

Another way to classify ATS is based on its purpose: Generic summarization provides a broad overview of the content by highlighting key ideas and main topics, making it useful for readers who seek a general understanding of a document or collection of texts, such as research papers or news articles. Query-based summarization focuses on extracting information tailored to specific user queries, such as summarizing research articles on "renewable energy trends" or addressing questions like "latest advancements in artificial intelligence." Lastly, domain-specific summarization targets specialized fields, summarizing content from areas like medical reports, legal documents, or financial analyses to ensure relevance and utility within a particular domain [6].

1.2 Problem Statement

The problem can be formulated as follows: Given an Arabic document D_{input}, represented as an ordered set of sentences $D_{input} = \{S_1, S_2, ..., S_n\}$, where each sentence S_i corresponds to the i-th sentence in the document and n represents the total number of sentences, with each sentence S_i further represented as a set of tokens $S_i = \{t_1, t_2, ..., t_m\}$, where t_k denotes the k-th token in S_i and m is the total number of tokens in that sentence. An automatic extractive text summarization system transforms D_{input} into an output summary $D_{output} = \{S_1, S_2, ..., S_k\}$, comprising selected sentences from D_{input}. The transformation aims to: (1) ensure D_{output} captures the essential information from D_{input}, and (2) limit the size of D_{output} to satisfy summary ratio.

1.3 Methodology

One widely used approach in extractive text summarization is graph-based methods. These methods represent a document as a graph, where nodes correspond to sentences, and edges denote the similarity between them. To identify sentences for inclusion in the summary, various ranking algorithms, such as Degree Centrality, TextRank, and LexRank, are commonly applied [7]. However, studies employing these techniques often face a significant limitation in capturing semantic meaning. This is primarily due to their reliance on traditional representation methods like TF-ISF or the omission of semantic features when calculating sentence similarity.

The proposed work is a graph-based approach to extractive automatic text summarization. The process begins with pre-processing, where classical methods, such as TF-ISF, involve steps like tokenization, normalization, stop-word removal, and stemming, while deep learning techniques focus on preserving semantic structure with minimal pre-processing. By incorporating semantic features, this work addresses the limitations of traditional methods, enabling a more nuanced understanding of sentence relationships.

A graph is then constructed with sentences as nodes and edges weighted by their semantic and structural relationships, utilizing features such as TF-ISF, FastText embeddings, and deep learning models like AraBERT. These semantic features enhance sentence representation, capturing contextual meaning and linguistic nuances often overlooked by classical techniques. Sentence similarity is calculated using cosine similarity and further refined by incorporating title relevance and key phrase scores, ensuring a more accurate representation of sentence connections.

To rank the sentences, the approach employs Weighted Degree Centrality, which considers statistical features like sentence position, length, cue words, and numerical data, or the adapted PageRank algorithm, named ST-Rank (Semantic TextRank), which integrates these features during score initialization. By combining semantic and structural features in these ranking techniques, the model enhances the quality of extracted summaries, ensuring they are both coherent and contextually relevant. Finally, the top-ranked sentences are selected to generate a concise and informative summary, effectively bridging traditional and modern methodologies to maximize performance.

1.4 Contributions

This research presents several key contributions to the field of automatic text summarization:

- **Integration of semantic and statistical features:** The proposed graph-based algorithm combines both semantic understanding and statistical measures to improve the accuracy of sentence selection. This hybrid approach ensures that the summary captures both the meaning and the importance of sentences.
- **Improved algorithm performance:** Compared to traditional methods such as TextRank and PageRank, the proposed technique shows faster convergence and higher-quality summaries. This improvement enhances the overall efficiency and reliability of the summarization process.
- **Superior results on the EASC dataset:** The method was evaluated on the well-known EASC dataset and achieved results that outperform existing state-of-the-art summarization techniques, demonstrating the effectiveness and robustness of the approach.
- **Development of a user-friendly web tool:** A practical web-based interface was created, allowing users to:
  - Upload '.txt' files or input text manually.
  - Specify the desired summary length or ratio.
  - Instantly generate and view the summary.

  This makes the system accessible and useful for a wide range of users.

1.5 Report Outline

The paper is organized as follows: Chapter two provides an overview of the most advanced Arabic extractive text summarization techniques, along with a discussion of related systems. Chapter three details the design and methodology of the proposed approach. In Chapter four, we outline the dataset, the evaluation metrics, the tools, and the anticipated experimental outcomes. Lastly, Chapter five concludes the paper by summarizing the findings and presenting future perspectives.

Chapter 2

Related Work

Contents

In this chapter, we revised some related studies in filed of automatic text summarization approaches. The first section talks about common approaches in text summarization including statistical-based methods, semantic-based methods, machine learning-based methods and optimized methods. The second section focuses on Graph-based methods in details.

2.1 Automatic Text Summarization Approaches

Automatic text summarization is a fast growing field with various approaches that aim to make useful and concise summaries from large text corpora. These approaches can be classified into different methods based on they process. In this section, different categories of ATS approaches will be discussed. First, statistical-based methods which rely on word frequency and co-occurrence patterns. Second, semantic-based techniques which focus on understanding the meaning and relationships between words. Third, Machine learning approaches, that uses supervised and unsupervised models to predict the most relevant content. Finally, optimized approaches combines advanced algorithms to improve summary quality by balancing some criteria.

2.1.1 Statistical-Based Methods

Statistical-based methods work on the extraction of the important sentences and words from the input documents based on statistical calculations of a set of features. These features can include the position of the sentence, the frequency of the sentence [8]. Statistical-based methods in automatic text summarizing systems play an essential role in the extraction of relevant information from huge text documents. These approaches include several methods for representing sentences. one method is Term Frequency-Inverse Document Frequency (TF-IDF). It calculates the weight of each term based on its occurrences compared with its occurrences in the entire corpus [9]. This statistical metric helps in identifying the unique terms in the document which leads to give more priority for the important sentences that will be included in the summary.

Hashem et al. [10] employed an improvement approach for email summarization using statistical-based methods. This approach focuses on extractive email summarization by selecting sentences to include in the summary based on statistical measures. The proposed approach applies sentence scoring, assigning each sentence in the email a numerical weight derived from seven important features. These features are derived from pre-processing steps such as segmentation, tokenization, stemming, and stop-word removal. The approach was evaluated using the Enron Sent Mail Sample dataset, demonstrating an improvement in summarization performance. It achieved a higher average similarity score with the reference (gold) summaries, with an average similarity value of 78.2

2.1.2 Semantic-Based Methods

Semantic-based methods for text summarization focus on producing meaningful and important summaries by understanding the meaning of words and phrases within a document. These methods use in understanding of language in addition to simple word matching to determine the connections and meanings between various text elements. The primary benefit of using semantic-based techniques is that they may produce summaries that are more accurate in relation of context, especially when analyzing complex topics.

Topic-Based Methods. In order to give a brief summary for a text, Topic-based methods in text summarizing focuses on finding the most important subjects or topics in a document. The method's concept is that by understanding a text's main ideas, it may collect the most important information for creating an accurate and meaningful summary. The basic idea of topic-based methods in text summarization is that documents often contain multiple topics. Latent Dirichlet Allocation (LDA) algorithms are employed to extract those topics from the text. The sentences that have a strong relationship with each topic are sorted into clusters when topics have been determined [11]. The main terms that represent the primary concepts related to a certain topic within the document can be found in each cluster. This method detects one or more topics for each sentence in the text. The objective is to cover the most information as possible to ensure the summary is clear and accurately combines up everything that was covered. The main benefit of this method is that it enhances the topics included in the original text, leading to the development of better, more useful summaries [12].

The main topics of a document are represented using various techniques in this method, such as measuring the importance of a word in relation to other texts (TF-IDF) or counting the frequency

with the terms appear (Term Frequency). Lexical chains, which link similar words, and topic word approaches, which identify key concepts and their importance, are two different methods. The key concept is to change the given text into a format that gives focus to the important points. Sentences are then ordered according to how closely they connect to these topics. Only the most important sentence is included in the summary. This enables the writing of a summary that clearly summarizes the key ideas in the text [13].

Liu et al. [14] developed an interactive system that allows users to view and analyze text based on specific topics. This system uses algorithms to detect and cluster themes within large text corpora, generating concise overviews focused on essential information for each topic. Using the full TREC (Text REtrieval Conference) dataset,this dataset contains a collection of news articles and is typically used for information retrieval and summarization tasks. They evaluated the system's performance, achieving an F-measure of 0.75, which indicates strong accuracy in retrieving and clustering relevant information. The visualizations facilitated users' understanding of complex topic interconnections, enhancing user engagement and comprehension. This study highlights the importance of visual tools in topic-based summarization, greatly improving users' ability to retrieve and analyze information effectively.

Latent Semantic Analysis (LSA) Method. One effective method for summarizing and analyzing text is Latent Semantic Analysis (LSA), which shows the hidden meaning of terms and words, without the need for previous training or outside knowledge, this method focuses on understanding the relationships between words depending on the situation in which they appear, enabling an increased understanding of the contents.

Latent Semantic Analysis (LSA) is based on the basic idea that words often appear together in sentence, which may help in our understanding of their meanings. For example, if we observe that a large number sentences related to "cats" are located in close range to sentences relating to "pets", LSA could assume that these topics are related. This method shows the importance of common words in understanding the meaning and relationships between sentences. Essentially, LSA may show hidden connections between various concepts in the text by looking at which words appear together [15] [16].

The building of a term-document matrix starts the LSA procedure. This matrix may be visualized as a table with words in each row and sentences in each column. The frequency of each word in each sentence is shown by the numbers in the table [15]. After this table is created, it is divided into smaller sections using a mathematical procedure known as Singular Value Decomposition (SVD). By Doing this, LSA is able to identify relationships between sentences and words, which improves understanding of their relationships. In the end, by focusing the most important connections, it helps in text summarization [17].

After the term-document matrix analysis, LSA scores every sentence according to its importance to the main concepts covered in the text. Next, the most importance word are selected to provide a straightforward, understood summary of the document [17]. In this way, readers may easily and quickly understand the main points of the text because the summary focuses on the importance ideas.

Mandal et al. [18] focused on text summarization using a Latent Semantic Analysis (LSA) approach. This method involved creating a word-document matrix to analyze sentence frequency within a document set. Singular Value Decomposition (SVD) reduced data dimensionality, revealing hidden semantic patterns. Sentences were scored based on their connection to core concepts, and those with higher scores formed the summary, reflecting key ideas. Using the DUC (Document Understanding Conference) dataset, the LSA-based approach achieved a Rouge-1 score of 0.62, underscoring its effectiveness in enhancing summary coherence and semantic relevance.

Semantic Role Labeling (SRL) Method. Semantic Role Labeling (SRL) is a natural language processing (NLP) technique used in sentence understanding. Using this method, people can analyze the sentence and determine the meaning of each word in relation to the verb or main action. In the sentence "Sarah gave Tom a gift", for example, SRL enables us to identify the action's topic (the gift), whose worker (Sarah), and the person who received it (Tom). Finding the roles of the words connected to the most important actions (predicates) in a sentence is the primary task of the SRL process. Finding out who is doing the action, what is being affected, and the meaning of the action represents the goals [19].

In SRL, every role is associated with a larger concept known as a "semantic frame" which contains the important features and interact of the entities that perform the activity. Two essential resources are needed to help with this process: FramNet and ProbBank. These materials offer a wide range of knowledge on all the roles that words may perform and their relationship to verbs. We can better understand sentences and their meaning by using SRL, which is helpful for many different kinds of applications including information extraction and text summarization [20].

An effective approach that improves the capacity to extract important information from long documents is text summarization using Semantic Role Labeling (SRL). Essential sentence parts, including "who did what to whom" and "under what conditions" are made easier to recognize by SRL and are essential for understanding the text's major concepts.

Mohamed et al. [21] present a text summarization approach combining Explicit Semantic Analysis (ESA) with Semantic Role Labeling (SRL). SRL analyzes the roles and relationships of words in sentences, enhancing semantic understanding, while ESA leverages external knowledge to improve text comprehension. Using the TAC (Text Analysis Conference) dataset, the method achieved a ROUGE-2 score of 0.54, demonstrating that this combined strategy outperformed traditional summarization methods. The summaries produced were more informative and aligned well with human judgments in clarity and relevance, showing the value of semantic analysis in summarization.

2.1.3 Machine Learning Methods

With its easy and strong techniques for producing short summaries, machine learning became known as an important part in modern text summarizing. In general, these methods can be classified as three groups: reinforcement learning, unsupervised learning, and supervised learning. Depending on the type of text and the end goal, each strategy has advantages and disadvantages of its own.

Supervised Learning. Supervised learning relies on a large amount of labeled data, where the system is trained to understand the relationship between input texts and their corresponding summaries [22]. The idea is to ”teach” the model by showing it examples of how summaries should look, so it can learn the patterns and apply them to new, unseen documents. The model uses algorithms to extract features from the text, such as sentence importance and key phrases, to generate summaries that align with the examples it was trained on.Models like recurrent neural networks (RNNs) and transformers, such as BERT, are often used because they can process sequences of text effectively. One big advantage here is that the results often closely match human-written summaries, as the model is trained to duplicate them. For instance, in a news summarization task, a supervised learning model might be trained on a dataset of news articles paired with headline summaries, learning to identify the most relevant information to include in the summary based on the input text [23]. However, supervised learning can be quite data-hungry—it requires extensive, high-quality datasets, which can be difficult and expensive to curate [24]. But once trained, these models can generate very accurate summaries.

Several researchers have expressed varied opinions on the effectiveness of supervised learning in text summarization [24] emphasized that supervised learning models, particularly neural network-based models, perform exceptionally well in generating coherent and fluent summaries, especially when a large dataset is available for training. However, Cheng et al. [25] pointed out the challenge of scalability, as supervised learning models heavily rely on vast amounts of labeled data, which is often expensive and time-consuming to produce.

Unsupervised Learning. In reverse, unsupervised learning does not require labeled datasets to function. The model looks for patterns in the data itself to determine the most significant passages in the text rather than depending on instances of input-output pairs. This approach uses a number of techniques that are good at identifying latent structures in the text, including clustering and topic modeling. For example, clustering algorithms might use co-occurrence patterns, word frequency, or semantic similarity to group phrases or paragraphs that are similar [26]. The model can produce a summary that captures the main ideas in the document by using these categories to determine which phrases are typical of the entire text.

One popular approach in unsupervised learning is to use clustering to select representative sentences from each cluster. By grouping similar sentences together, the model can choose a sentence from each cluster that best represents the entire group, effectively reducing redundancy and emphasizing diversity. This method allows the summarization process to be more efficient, especially when dealing with large datasets. Topic modeling techniques, such as Latent Dirichlet Allocation (LDA), are also frequently used to uncover the major topics within the text, and these topics can then be used to guide the selection of sentences for the summary [27]. The benefit here is flexibility, as unsupervised models can process any type of text without needing pre-prepared summaries for training, making them particularly useful in domains where labeled data is scarce or unavailable.

However, unsupervised learning methods have their limitations. While they are highly adaptable and do not require extensive training data, they often struggle to achieve the same level of fluency or coherence as supervised approaches [28] noted that while unsupervised techniques

such as LexRank can generate decent summaries, they often lack the natural flow and precision found in human-written summaries or those produced by supervised learning models. This can result in summaries that are less cohesive or omit crucial details. Still, for many tasks, especially where the availability of labeled data is a constraint, unsupervised methods provide a viable solution, delivering satisfactory results in an efficient manner.

Reinforcement Learning. Reinforcement learning (RL) presents a distinct approach to text summarization, fundamentally different from both supervised and unsupervised methods. In RL, the model learns by trial and error, continually adjusting its behavior to achieve specific goals. The process involves generating summaries and receiving feedback in the form of a reward signal, which helps the model understand how well its summary meets desired objectives, such as fluency, relevance, or conciseness. Through this iterative process, the model is able to refine its output, gradually improving by making decisions that maximize the rewards over time. Unlike supervised methods, where the model relies on labeled data to learn patterns, RL encourages the system to actively explore different strategies, fostering creativity in how it summarizes text [29].

The power of RL to balance many multiple objectives is one of its special advantages in text summarization. For instance, RL can be adjusted to balance the trade-off between extractive and abstractive summaries, as well as to improve both speed and accuracy. Whereas abstractive summarizing creates new phrases with the same meaning, extractive summarization chooses sentences straight from the original text. By varying the proportion of each technique employed based on the training objectives, reinforcement learning offers more flexibility. This flexibility is particularly useful in complex tasks, such as news summarization or legal document analysis, where the system needs to balance detail and brevity [30]. The ability to fine-tune these trade-offs gives RL a significant advantage over more rigid, traditional summarization methods.

However, the challenge of defining the reward function is one of the key obstacles in applying reinforcement learning to summarization tasks. If the reward function is poorly designed, it may incentivize the wrong aspects of a summary, leading to outputs that appear superficially effective but fail to capture the deeper meaning or nuances of the text. For instance, if the reward is overly focused on fluency, the model might generate grammatically correct summaries that lack key information [30] emphasize that achieving the right balance in reward function design is crucial for ensuring high-quality outputs. Despite these challenges, RL offers a powerful framework for creating more nuanced, goal-driven summaries, making it particularly useful in areas where traditional methods struggle to account for multiple objectives.

2.1.4 Optimized Methods

Optimization methods in ATS systems plays an important role in identifying and selecting the core sentences of the text in order to represent the summary while keeping it concise. The optimization methods are used to determine which sentences that best represent the document/s. This process is done through mathematical models to evaluate the importance of a sentence. The importance of a sentence is determined according to some criteria including informativeness, relevance, redundancy, etc. Moreover, optimization methods aim to enhance the computational efficiency of the summarizing process in large documents. This objective is essential in real-time applications. Optimization methods in ATS systems are classified into two types

single-objective and multi-objective optimization methods [31].

Single-Objective Optimization. Single-objective optimization methods focus only on one primary goal. Single-objective optimization in ATS means the primary goal is optimizing criterion or objective function, such as relevance, informativeness, or coherence of the generated summary [31]. Single-objective optimizations are simple and computationally less, but they may not always balance multiple desirable qualities [32] explain wide range of nature-inspired algorithms used for solving single-objective optimization problems. In this section, three algorithms will be discussed.

1- Genetic Algorithm (GA) Genetic algorithm is a nature-inspired optimization technique. Individual people in the population represent potential solutions for an optimization problem. They can be the solution by improving over generations through fitness, selection, crossover, and mutation. recent improvements were made on GA (2019-2023) including hybrid approaches that combines GA with other algorithms (e.g., PSO or DE).

2- Particle Swarm Optimization (PSO) The PSO algorithm is inspired from the social behavior of bird flocks or fish schools. In this algorithm, the individuals or the particles go through the search space. They influence by their own best-known position and the best-known positions of other particles. Recent improvements were made on this algorithm (2019-2023) including enhancement of parameters adaptation such as inertia weight, cognitive, and social parameters. Also new hybrid PSO methods that integrate other optimization techniques to handle specific problem types like large-scale optimization or dynamic environments.

3- Ant Colony Optimization (ACO) ACO algorithm is inspired by the behavior of ants searching for food, where they deposit pheromones to mark paths to food sources, and subsequent ants are more likely to follow stronger pheromone trails, reinforcing successful paths. This algorithm represents this behavior to solve optimization problems. Recent improvements were made on ACO algorithm including adaptive pheromone updating rules and combination with other algorithms to enhance the search for global optima.

Multi-Objective Optimization. Multi-objective optimization methods aim to simultaneously enhance conflict objectives. in ATS for example, while generating the summary there should be a balance between maximizing relevance of the summary with the documents and minimizing the redundancy from the summary [18].

Xie et al.[33] introduces a multi-objective optimization framework that aims to enhance Automatic Text Summarization (ATS) by balancing several objectives, such as maximizing content or topic coverage in the summary while minimizing redundancy. This framework has used the DUC2004 dataset with the ROUGE evaluation metric, specifically emphasis on ROUGE-1 Recall as a submeasure, achieving a value of 0.556. The methodology employs an optimization algorithm to select sentences from the input document(s) based on these objectives. The results of this experiment indicate that the approach produces more coherent and informative summaries compared to traditional methods.

2.2 Graph-Based Methods

Graph-based methods in ATS follow a set of common steps to create summaries using mathematical graph theory. First, preprocessing is performed, such as text normalization, tokenization, and removing stop words. Second, a centrality graph is constructed where sentences are represented as nodes, and edges represent relationships between those sentences. These relationships are usually based on measures like cosine similarity or mutual information. Third, a scoring mechanism is applied to assign scores to the sentences, reflecting their importance in the graph. Finally, the sentences are ranked, and the top-ranked ones are selected to form the summary [34]. This section underlies various methods such as Degree-centrality, TextRank, and LexRank, each of which employs unique strategies for scoring and ranking sentences. The following sections explains the specifics of these approaches.

2.2.1 Degree-Centrality Method

The Degree-Centrality Method is a graph-based approach widely used in extractive text summarization. It evaluates the significance of sentences by analyzing their connectivity in a graphical representation of the text. The basic idea is that the most important sentences contain higher quality information and are so important to the summary have many connections to the other sentences in the graphical representation, and this is what give them the importance than the other sentences in the graphical representation. This method is particularly effective for single-document summarization tasks, where the goal is to create concise, informative representations of the original text[35].

Degree-centrality focuses on the representation of a document as a graph. Nodes represent sentences, while edges represent semantic or lexical similarity between sentences. Each node's degree, which is the number of connections it has with other nodes, determines its centrality score. Sentences with higher degrees are prioritized, under the rationale that they encapsulate broader context[36]. Formally, Let $G(V, E)$ represent a graph where V is the set of nodes (sentences) and E is the set of edges (similarities). The degree-centrality of a node $v \in V$ is defined as:

$$\mathrm{C}(v) = \sum_{u \in V} w(u, v) \tag{2.1}$$

where $w(u, v)$ is the weight of the edge between nodes u and v. A threshold can be applied to $w(u, v)$ to determine the presence of an edge. Degree-centrality provides a measure of a sentence's importance relative to others in the graph.

Ghaleb Algaphari et al.[35] In their work "Text Summarization Using Centrality Concept", the authors applied a degree-based centrality method to summarize Arabic texts. The study specifically focused on Arabic news articles, utilizing cosine similarity to measure the semantic similarity between sentences. The proposed algorithm represents the text as a fully connected graph, where each sentence serves as a node, and edges reflect lexical similarities. Degree centrality was employed to rank sentences based on their importance in the similarity graph, using a threshold to balance precision and coverage. The algorithm was evaluated on the Arabic Newswire-a corpus provided by the Linguistic Data Consortium (LDC), comprising 100 docu-

ments divided into five topics. The evaluation combined manual grading and ROUGE metrics, achieving an average ROUGE-1 score of 0.683, ROUGE-2 score of 0.5308, and a human grading score of 3.89 out of 5. The results highlighted the efficacy of degree centrality in generating coherent and relevant summaries, especially for low-resource languages like Arabic. The study demonstrated that degree centrality could effectively summarize documents with rich linguistic structures while maintaining scalability and adaptability.

Hariharan et al. [37] proposed enhancements to graph-based methods for single-document summarization by refining the application of centrality measures. They specifically updated the Degree-Centrality method to improve its effectiveness in ranking sentences by considering the structure of the sentence graph and adjusting sentence weights based on their connectivity. For evaluation, the authors used the DUC-2002 dataset and applied the ROUGE-1 measure to assess summarization quality. Their enhanced method achieved a ROUGE-1 score of 0.392, indicating an improvement in summarization performance over traditional graph-based methods.

2.2.2 TextRank Method

Google developed PageRank to solve a key issue they faced with their search engine for the World Wide Web (Brin and Page, 1998) [38]. When users submitted a search query, the engine would return a vast array of web pages that matched the search terms exactly. However, Google needed a way to differentiate between pages based on their importance and relevance. To achieve this, they introduced PageRank, a scoring system that analyzed the web's link structure to determine a page's significance.

PageRank operates based on the principle of analyzing the link structure of the web to determine the importance of individual pages. It models the web as a directed graph where web pages are represented as nodes (vertices) and hyperlinks as directed edges. Formally, let $G = (V, E)$ be a directed graph, where V is the set of vertices and E is the set of edges, with $E \subseteq V \times V$. For a given vertex V_i, let $\text{In}(V_i)$ represent the set of vertices that point to V_i (predecessors), and let $\text{Out}(V_i)$ represent the set of vertices that V_i points to (successors). The score of a vertex V_i is defined as follows: [38]

$$\text{S}(V_i) = (1 - d) + d \sum_{j \in \text{In}(V_i)} \frac{1}{|\text{Out}(V_j)|} S(V_j) \tag{2.2}$$

In this equation, d represents a damping factor that can be adjusted between 0 and 1. It reflects the probability of transitioning from one vertex to a randomly selected vertex in the graph. This graph-based ranking algorithm embodies the "random surfer model" commonly associated with web browsing, where a user clicks on links randomly with a probability of d and jumps to a completely new page with a probability of 1−d [39].

Similarly, we can apply this approach to lexical or semantic graphs derived from natural language documents, resulting in a graph-based ranking model known as TextRank. This model is utilized in various natural language processing tasks, leveraging knowledge from the entire text to inform local ranking and selection decisions. TextRank is a content extraction algorithm that models texts as graphs to facilitate the extraction of sentences and keywords. It employs the PageRank algorithm to assign rankings to these sentences or keywords based on their importance

within the text. This method allows for effective summarization and keyword identification by leveraging the relationships between text elements [39]. Table 2.1 shows a breif compassion between PageRank and TextRank.

Feature	**PageRank**	**TextRank**
Node Representation	Web pages as nodes	Words or sentences as nodes
Edge Definition	Hyperlinks between pages	Word co-occurrence or sentence similarity
Edge Weights	Equal weight for all links	Weighted by co-occurrence frequency or similarity score
Application Domain	Web page ranking in search engines	NLP tasks like keyword extraction and summarization
Supervision	Unsupervised	Unsupervised

Table 2.1: Comparison between PageRank and TextRank algorithms

Al-Khassawneh et al. [7] propose an extractive summarization method specifically designed for Arabic text using a graph-based approach. The methodology focuses on constructing a graph where each sentence in the document is represented as a node, with edges weighted based on a customized similarity formula for Arabic text. This formula includes linguistic features like root-based word similarity, word order, and sentence length to enhance the weighting of graph edges. The modified TextRank algorithm processes the graph to determine the most significant sentences for summarization. The study involved several steps to enhance the summarization process: preprocessing the Arabic text through stop-word removal, stemming, and normalization, constructing a graph with enhanced similarity measures, iteratively scoring sentences with the modified TextRank, reducing redundancy to ensure diversity and generating summaries by selecting sentences with the highest scores. The method was evaluated using the Essex Arabic Summary Corpus (EASC) and assessed with the ROUGE metric, achieving an average ROUGE score of 0.52, which demonstrated notable improvement over baseline methods.

Similar to TextRank, Elbarougy et al. [40] introduce a method for summarizing Arabic text that improves the PageRank algorithm. They use the Al-Khalil morphological analyzer to extract nouns which are key to understanding Arabic sentences. Each sentence in the text is treated as a node in a graph, and sentences are connected based on their similarity. Instead of giving all sentences the same starting score, the method assigns a scores equals to the number of nouns that the sentence has since these are considered more informative. The process involves three main steps: preprocessing the text by removing stop words, constructing a graph by connecting sentences based on their similarity using cosine similarity, and applying the modified PageRank algorithm to rank the sentences. The method generates summaries by selecting top-ranked sentences, avoiding repetition by checking for overlaps. When tested on the EASC dataset, it has achieved precision, recall, and F-measure scores of 0.687, 0.729, and 0.679, respectively. The best results were achieved after 10,000 iterations.

Another study that is similar to TextRank is Bichi et al. [41]. They proposed a new method to summarize Hausa text by improving the PageRank algorithm. Their modification involves changing how the initial scores of graph nodes are calculated. Instead of using uniform or

frequency-based initial scores, they used the normalized common bigram counts between sentences. This means they prioritized sentences that share frequently occurring two-word combinations with others, improving the relevance of selected sentences in the summary. The process involved several steps. First, collecting and preprocessing the text, including tokenization, stopword removal, and stemming. Second, creating a graph where sentences are nodes, and edges are weighted by their cosine similarity. Finally, applying the modified PageRank algorithm to rank and select the most relevant sentences. They tested their method on the Hausa Summarization Evaluation Dataset, which includes 113 news articles. Using the ROUGE evaluation toolkit, their method outperformed others, improving on TextRank by 2.1

Gulati et al. [42] introduced a new way to improve extractive text summarization by combining TextRank with the BM25+ model. They changed the original TextRank algorithm by adding a normalized similarity matrix that merges the strengths of both models to better rank sentences for summaries. The method works by creating a graph where sentences are treated as nodes, and the edges represent how similar each sentence is to the others. Then, the most important sentences, based on their rank, are selected to form a summary. The authors tested their approach using a dataset of 75 blogs published in 2018 on the Medium platform, covering 25 domains such as technology, science, and entertainment. Each article was paired with a manually generated expert summary. The articles ranged from 1,003 to 5,185 words in length, with an average of 100 sentences per article. The authors used ROUGE scores (ROUGE-1, ROUGE-2, and ROUGE-L) and measured recall, precision, and F1 scores. The results showed that their modified method outperformed traditional approaches, including TextRank, TF-IDF, LCS, and BM25+, with improvements in all evaluation metrics. Specifically, their approach achieved a recall of 0.7765, precision of 0.7398, and an F1 score of 0.7537 for ROUGE-1. For ROUGE-2, the recall was 0.6749, precision 0.6437, and F1 score 0.6554. The ROUGE-L scores were 0.7486 for recall, 0.7153 for precision, and 0.7283 for F1, indicating its effectiveness across various domains.

2.2.3 LexRank Method

LexRank is a graph-based method used in natural language processing (NLP) for text summarization. It identifies the most important sentences in a document or across multiple documents by representing sentences as vertices in a graph. The edges between sentences indicate their similarity, based on lexical content rather than hyperlinks, similar to how PageRank ranks web pages. LexRank calculates sentence importance by analyzing their connections, making it especially effective for multi-document summarization, where information from several sources is combined into a clear and concise summary [43].

LexRank operates on the principle of computing the centrality of sentences in a text based on their similarity to other sentences. It models the text as a graph where sentences are represented as nodes (vertices) and the similarity between sentences as weighted edges. Formally, let $G = (V, E)$ be a weighted graph, where V is the set of vertices (sentences) and E is the set of weighted edges, with $E \subseteq V \times V$. For a given vertex V_i, let $\text{In}(V_i)$ represent the set of vertices that point to V_i (predecessors), and let $\text{Out}(V_i)$ represent the set of vertices that V_i points to (successors). The centrality score of a vertex V_i is defined as follows [28]:

$$\text{S}(V_i) = \frac{d}{N} + (1 - d) \sum_{j \in \text{In}(V_i)} \frac{\text{sim}(V_i, V_j)}{\sum_{k \in \text{Out}(V_j)} \text{sim}(V_j, V_k)} S(V_j) \tag{2.3}$$

Explanation of Symbols:

- $S(V_i)$: The centrality score or importance of the sentence V_i.
- d: The damping factor, typically set to 0.85, similar to PageRank, to model the probability of jumping to another sentence.
- N: The total number of vertices (sentences) in the graph.
- V: The set of vertices (sentences) in the graph.
- E: The set of edges representing the similarities between sentences.
- In(V_i): The set of sentences that point to sentence V_i (predecessors).
- Out(V_i): The set of sentences that sentence V_i points to (successors).
- sim(V_i, V_j): The similarity score between sentences V_i and V_j.

Erkan et al. [28] proposed an innovative method for extractive text summarization called LexRank. This method is based on the PageRank algorithm but specifically designed for summarization tasks by using sentence-level lexical similarity. The key innovation in LexRank lies in the use of a graph-based centrality approach, where sentences are represented as nodes, and edges are weighted by the cosine similarity between sentences. They introduced two versions of the algorithm: Continuous LexRank, which uses fully connected graphs with proportional weights, and Thresholded LexRank, which reduces the graph's density by only connecting sentences with similarity above a specified threshold. The authors did not modify an existing implementation of LexRank but developed it as a distinct method for summarization.For evaluation, they used the Document Understanding Conference (DUC) 2003 and DUC 2004 datasets, which are benchmark datasets for text summarization. They measured the performance using the ROUGE evaluation metric, particularly ROUGE-1, ROUGE-2, and ROUGE-SU4 scores, which compare the overlap of n-grams and skip-grams between the generated summaries and human-written reference summaries.LexRank achieved strong performance in comparison to other summarization methods. For the DUC 2003 and DUC 2004 datasets, LexRank achieved ROUGE-2 scores of approximately 0.41 and 0.42, respectively. These results were competitive with other graph-based methods like TextRank. Additionally, the study showed that using a low threshold (0.1) in the Thresholded LexRank version preserved key information while maintaining summary quality, further enhancing the approach's effectiveness.

Deo et al. [44] in this paper the authors used LexRank in its original form, without any modifications. They compared LexRank with other algorithms, including TextRank and Latent Semantic Analysis (LSA), to evaluate their performance in extractive text summarization. The study applied these algorithms to blog datasets from various categories, including social science, health, and agriculture. The results showed that each algorithm had different strengths depending on the evaluation parameters, such as cosine similarity, unit value, and recall value. For the social science category, LexRank achieved a cosine value of 0.7402, a unit value of 0.3435, and a recall value of 0.1971. In the health category, LexRank performed with a cosine value of 0.6984, a unit value of 0.1765, and recall value of 0.2928. For the agriculture category,

LexRank showed a cosine value of 0.7574, a unit value of 0.3435, and recall value of 0.2754. The authors did not mention any adjustments made to LexRank but instead used it as a standard algorithm for comparison with the other methods.

Table 2.2 provides a summary of studies on text summarization techniques, comparing approaches like Degree-Centrality, TextRank, and LexRank. The table highlights the features used, such as cosine similarity and sentence lengths, the datasets applied, including EASC and DUC, and evaluation metrics like precision, recall, and ROUGE scores to assess the effectiveness of these methods.

Study Title	**Approach**	**Features used**	**Dataset**	**Evaluation Results**
Text Summarization Using Centrality Concept [35]	Degree-Centralty	Sentence Centrality,Cosine Similarity,TF-IDF Score,Sentence-to-Sentence Similarity and Title Similarity	Arabic NEWSWIRE-a corpus	Average Precision: 0.6463 , Average Recall: 0.7199 , ROUGE-1: 0.683 ,ROUGE-2: 0.5308 , ROUGE-3: 0.375 and The average F1-measure for the dataset is 0.71988
Extractive Arabic Text Summarization-Graph-Based Approach [7]	TextRank	Root-based word similarity, word order, and sentence lengths	Essex Arabic Summary Corpus (EASC)	Average ROUGE score of 0.52
Extractive Arabic Text Summarization Using Modified PageRank Algorithm [40]	TextRank	Number of nouns in a sentence and cosine similarity for sentence connections	Essex Arabic Summary Corpus (EASC)	Precision: 0.687, Recall: 0.729 and F-measure: 0.679
Graph-based extractive text summarization method for Hausa text [41]	TextRank	Normalized common bigram counts between sentences and cosine similarity for sentence connections	Hausa Summarization Evaluation Dataset (113 news articles)	TextRank by 2.1%, LexRank by 12.3% and BM25 by 17.4% (based on ROUGE evaluation)
Extractive Article Summarization Using Integrated TextRank and BM25+ Algorithm [42]	TextRank	Normalized similarity matrix combining TextRank and BM25+	75 blogs published in 2018 on Medium, spanning 25 domains. Articles ranged from 1,003 to 5,185 words, averaging 100 sentences each, paired with manually generated expert summaries	ROUGE-1: Recall: 0.7765, Precision: 0.7398 and F1 Score: 0.7537. ROUGE-2: Recall: 0.6749, Precision: 0.6437 and F1 Score: 0.6554. ROUGE-L: Recall: 0.7486, Precision: 0.7153 and F1 Score: 0.7283
An Approach for Combining Multiple Weighting Schemes and Ranking Methods in Graph-Based Multi-Document Summarization [45]	PageRank (enhanced with combined weighting schemes)	Cosine similarity, Jaccard similarity, Euclidean distance (combined using harmonic mean)	DUC 2003, DUC 2004	DUC 2004: ROUGE-1: 1.53% improvement over TextRank; ROUGE-2: up to 0.96% improvement over TextRank; Outperformed GRU+GCN by 1.07%
LexRank: Graph-based Lexical Centrality as Salience in Text Summarization [28]	LexRank	sentence-level lexical similarity (measured using cosine similarity) and graph-based centrality for ranking sentences	Document Understanding Conference (DUC) 2003 and DUC 2004	For the DUC 2003 and DUC 2004 datasets, LexRank achieved ROUGE-2 scores of approximately 0.41 and 0.42, respectively.

Table 2.2: Summary of Studies on Text Summarization

Chapter 3

Proposed Work

Contents

The proposed work consists of four main stages as shown in Figure 3.1. It begins with text pre-processing to clean and structure the input. For traditional methods like TF-ISF, techniques such as tokenization, normalization, stop-word removal, and stemming are applied, while deep learning approaches, focus on preserving semantic structure with minimal pre-processing. A graph is then constructed, where sentences are represented as nodes and weighted edges capture their associations based on semantic and structural relationships, using three methods TF-ISF, FastText embeddings, and deep learning model. Sentence similarity is calculated using cosine similarity, refined with title relevance and key phrase scores to enhance sentence importance. Sentences then are ranked using Weighted Degree Centrality, which considers four statistical features: sentence position, length, cue words and numarical data the sentence it has. Additionally, an adapted PageRank algorithm named as ST-Rank (Semantic TextRank) used as another ranking method. It initializes sentences' scores with a combination of the statistical features mentioned earlier. Finally, the most relevant sentences are selected based on their ranks to generate a concise and informative summary.

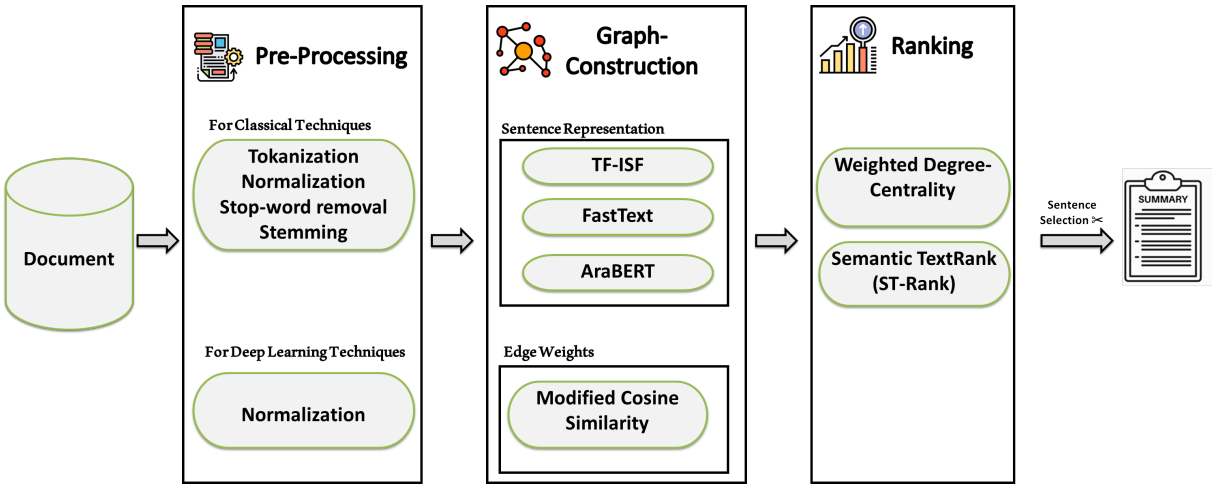

Figure 3.1: Proposed approach

3.1 Text Pre-Processing

This stage is the starting point for almost all summarization methods. Its main goal is to prepare the input text document for the following stages by converting it into a consistent and standardized format. This ensures that the document is ready for further processing and analysis. The specific pre-processing techniques used can vary depending on the type of sentence representation being applied. The upcoming sections provide an overview of the pre-processing methods used with classical sentence representation techniques, as well as those designed for deep learning-based sentence representation methods.

3.1.1 Text Pre-Processing for Classical Sentence Representation

In the context of classical sentence representation, pre-processing plays a crucial role in preparing Arabic text for further analysis using models like TF-ISF. The main goal of this stage is to clean and structure the text, making it easier for the model to extract meaningful features. Common techniques include tokenization, normalization, stop-word removal, punctuation removal, and stemming. These techniques, when applied correctly, can significantly improve the performance of text summarization tasks that use TF-IDF [46] [47]. Figure 3.2 provides a sample output demonstrating the sequential application of these text pre-processing methods.

Tokenization. Text preprocessing begins with the tokenization process. It breaks the input documents into units at various levels to simplify access to different parts of the text. These units can include paragraphs, sentences, tokens, numbers, or any other relevant unit [48].

Normalization. Normalization in Arabic text processing aims to standardize text variations. This work addresses the following cases: First, unifies certain letters that appear in various forms within the same word. For example, the letters "ء" (hamza), "آ" (alef mad), "أ" (alef with hamza on top), "ؤ" (hamza on waw), "إ" (alef with hamza at the bottom), and "ئ" (hamza on ya) are all normalized to "ا" (alef). Similarly, "ى" is normalized to "ي," and "ة" is converted to "ه." Second, diacritics such as ً ٌ ٍ َ ُ ِ are removed because they make a set of variations for the same term, so they will not be useful for tasks like root extraction or text classification. Third,

letters with the ”ّ” (shadda) symbol are duplicated, as the symbol plays a role in root extraction, and its removal can alter the meaning of the words. Fourth, removing ”tatweel” which is streching character. Lastly, remove redundant spaces. This process helps reduce variability in the text, making it more suitable for computational analysis while preserving essential linguistic information [49].

Stop-Word Removal. Stop words, such as pronouns, prepositions, and conjunctions, are commonly occurring words in text that are essential for sentence structure but carry little meaningful information [50]. Removing these words during pre-processing helps focus on the more informative terms that contribute to the document's core content. This step is particularly important because many computations, like word frequency analysis, rely on the presence of significant words. By eliminating stop words, these calculations become more precise and relevant, which will enhance the effectiveness of subsequent text analysis.

Stemming. Stemming refers to the process of stripping affixes, such as prefixes, infixes, and suffixes, from words to obtain their root form. This reduces different inflected forms of a word, which share the same meaning, to a single root or stem, making the feature space more consistent. By applying stemming techniques, the dependence on specific word forms is minimized, and the feature size is reduced, which enhances the performance of the model. In Arabic, common stemming techniques include the root-based approach and the light stemming approach [49].

The root stemmer aims to extract the morphological root of a word by removing all prefixes, suffixes, and infixes, thereby reducing the word to its core root. This approach is particularly suited for highly inflectional languages like Arabic, where many derived forms can share a common root. However, it may lead to over-stemming in some cases, resulting in the loss of important semantic distinctions [51]. In contrast, the light stemmer focuses on reducing words by removing affixes without attempting to identify the morphological root, preserving more of the word's structure. This method strikes a balance between reducing redundancy and maintaining semantic integrity [52]. The root-based stemming approach is a robust choice for classical sentence representation techniques like TF-ISF, as it provides a more compact and semantically meaningful feature space.Khurshid et al. [53] highlight that root stemmers often outperform light stemmers in traditional sentence representation tasks due to their ability to capture shared semantic roots effectively. Therefor, the root stemmer will be used in this work for this pre-processing stage.

3.1.2 Text Pre-Processing for Deep Learning Sentence Representation

Preprocessing techniques used with deep learning sentence representation methods share similarities with those used in classical sentence representation approaches, but there are notable differences. In deep learning, pre-processing typically excludes tokenization, as deep learning models are equipped with embedded tokenizers designed specifically for their architectures. Additionally, stemming and stop-word removal are generally avoided because these processes can alter the semantics of sentences, which deep learning models rely on to capture meaning effectively. Instead, common preprocessing steps include normalizing the text by removing diacritics, elongations, and special characters, converting text to lowercase, removing tatweel and

Figure 3.2: Sample output of the text preprocessing methods proceeded in sequence.

handling punctuation consistently. These steps aim to reduce noise in the input data while preserving the semantic structure that deep learning models depend on [54].

3.2 Graph Construction

In text summarization, a graph is a structured representation of relationships among elements within a document. It consists of nodes, which correspond to sentences from the document, and edges, which represent the similarity between pairs of sentences.[7] The graph is often modeled as fully connected, meaning every sentence node is linked to every other nodes. In this work, nodes are represented using three approaches. Term Frequency Inverse Document Frequency (TF-IDF), FastText and deep learning approach. Connections are weighted based on the degree of similarity between the sentences, calculated using cosine similarity, key phrase matching and sentences relevancy with the title. This approach allows the graph to capture the semantic and structural relationships among sentences effectively. Once constructed, the graph can be analyzed using algorithms like PageRank or its variations, which rank sentences based on their importance and connectivity within the graph.

3.2.1 Sentence Representatio

Sentences in a graph can be represented using different techniques, including classical methods like Term Frequency-Inverse Sentence Frequency (TF-ISF), modern approaches such as word embeddings like FastText, and advanced deep learning models like AraBERT. These techniques

have been explored and compared to assess their performance, providing valuable insights into their effectiveness for graph-based text representations. A detailed comparison of these techniques, highlighting their descriptions, advantages, disadvantages, and use cases, is presented in Table 3.1.

TF-ISF. TF-ISF is a numerical statistic that measures the relevance of a word in a sentence. It is often used in NLP to indicate a term's relevance to a text or a collection of sentences. The TF-ISF technique considers two key factors: the frequency of a word within a sentence (TF) and the frequency of the word across the entire corpus (ISF) [4]. TF-ISF assigns a weight to each phrase based on its frequency in the sentence compared to its occurrence across the full corpus. This statistical metric prioritizes essential data for a summary by focusing on sentence-specific phrases. TF-ISF can successfully detect and highlight key phrases in news stories, resulting in more useful and concise summaries [4]. Term frequency shows how often a term t is used in a particular sentence s:

$$\text{TF}(t, s) = \frac{\text{number of occurrences of } t \text{ in } s}{\text{total number of terms in } s} \tag{3.1}$$

Term frequency shows how often a term t is used in a particular sentence S:

$$\text{ISF}(t, S) = \log_{10}\left(\frac{\text{number of sentences in corpus } S}{\text{number of sentences in corpus } S \text{ containing } t}\right) \tag{3.2}$$

Combining the TF and ISF of a term, you develop the composite weight of terms in a corpus:

$$\text{TF-ISF}(t, S) = \text{TF}(t, s) \times \text{ISF}(t, S) \tag{3.3}$$

There are some disadvantages to using TF-ISF, including calculating sentence similarity directly in the word-count space, which can be slow with big vocabularies. TF-ISF does not take into account the semantic meaning of words. Also, it cannot understand the context of words and determine their meaning. Furthermore, it fails to define the semantic relationships between words [55].

FastText. FastText is languge model that simplifies text processing tasks such as classification and word representation. It translates words into numerical formats (word embeddings) by breaking them down into smaller components known as character n-grams. This strategy allows FastText to handle difficult or unknown phrases more effectively than older solutions. It recognizes both the meaning (semantics) and structure (syntax) of words, making it suitable for sentiment analysis and text categorization. FastText also allows for sentence-level representation by mixing word embeddings to determine the overall meaning of sentences [56]. FastText provides significant benefits over older approaches such as TF-ISF. While TF-ISF focuses mainly on word frequency and relevance within a sentence , it doesn't account for word meaning or relationships. FastText, on the other hand, learns words' contexts and semantics, allowing it to understand related words and their usage. FastText additionally handles unusual and out-of-vocabulary terms using a subword-based technique, whereas TF-ISF treats such words as totally new features, resulting in sparse and less informative representations. [57]

FastText has a few disadvantages as well. It can be computationally expensive and memory challenging particularly when dealing with huge datasets and vocabularies, because it employs n-gram algorithms. In limited resources contexts, higher memory use can be a problem. Also, FastText may not perform well in highly specialized fields without fine-tuned using domain-specific data. Unlike TF-ISF, which clearly shows essential words, FastText employs strong vectors that are more difficult to understand.[58]

AraBERT. is a transformer-based language model developed just for Arabic, depending on the BERT architecture to take into account the intricacies of Arabic grammar and syntax. AraBERT extracts complex contextual representations of words and phrases through pre-training on large Arabic text corpora, allowing it to recognize hidden meanings in the language [59]. AraBERT produces embeddings that capture both the semantic and syntactic components of a sentence, allowing for a thorough understanding of its meaning. This is accomplished by its bidirectional encoding, which includes the context of both preceding and subsequent words, resulting in more accurate and understandable phrase representations [59].

When compared to previous methods like TF-ISF and word embeddings like FastText, AraBERT provides considerable benefits by resolving their limitations. TF-ISF is based entirely on word frequency and lacks the ability to capture semantic links between words, limiting its efficiency in comprehending context. FastText improves on this by taking into account subword information, which allows it to handle out-of-vocabulary terms and morphological variants more well than TF-ISF.However, FastText falls short of capturing deep context connected since it produces static embeddings that remain constant independent of context. In contrast, AraBERT creates dynamic, context-dependent embeddings that reflect the meaning of words in their current context, allowing for a more complex understanding of language. This capacity makes AraBERT especially useful for jobs such as text summarization, where capturing complex meanings and context ambiguities is critical. [60]

In our implementation, we leveraged the ”aubmindlab/bert-base-arabertv2” model and utilized its internal hidden states to generate more semantically rich sentence representations. Instead of relying on the default output, we applied advanced pooling strategies, including mean pooling over the last two hidden layers and a weighted aggregation of the last four layers, where deeper layers were assigned higher weights. These strategies were designed to enhance the semantic quality of the sentence embeddings by combining contextual cues from multiple levels of the network. Each embedding was further normalized to ensure consistency in similarity computations across the graph. This pooling flexibility allowed us to experiment with various contextual depths and select the strategy that best aligns with the summarization task. The same approach was also applied to the title sentence to maintain coherence in semantic comparisons.

3.2.2 Edge Weights

Edges between nodes in the constructed graph for the text are representing the similarity between sentences. The measure used for calculating the similarity between sentences is Cosine similarity, which is a metric used to measure the similarity between two non-zero vectors in an inner product space, quantifying how closely they align regardless of their magnitude. It calculates the cosine of the angle between the vectors, with values ranging from -1 (indicating

Aspect	TF-ISF	FastText	AraBERT
Description	Measures word relevance based on frequency within a sentence (TF) and across a corpus (ISF).	Represents words as embeddings using character n-grams, capturing semantics and syntax.	Transformer-based model for Arabic, capturing context and semantics using BERT architecture.
Advantages	Simple and interpretable, highlights key phrases in text and effective for frequency-based tasks.	Handles rare and out-of-vocabulary terms, captures word semantics and syntax and suitable for sentiment analysis and text classification.	Captures deep contextual meanings, handles Arabic grammar and syntax effectively and generates dynamic, context-dependent embeddings.
Disadvantages	Ignores semantic meaning and word relationships, slow with large vocabularies and limited to word-count space.	Computationally expensive, memory-intensive for large datasets and less interpretable due to dense vectors.	Requires significant computational resources, needs domain-specific fine-tuning for specialized tasks and complex to implement and train.
Use Cases	Text summarization and keyword extraction.	Sentiment analysis and text classification.	Arabic text summarization and contextual understanding of Arabic text.

Table 3.1: Comparison of Sentence Representation Techniques

completely opposite vectors) to 1 (indicating identical vectors), and 0 signifying orthogonality. The formula for cosine similarity between two vectors A and B is:

$$\text{cosine similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.4}$$

Where $\mathbf{A} \cdot \mathbf{B}$ is the dot product of vectors **A** and **B**, $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ are the magnitudes (Euclidean norms) of vectors **A** and **B**, respectively. A_i and B_i are the components of vectors **A** and **B** [61].

Cosine similarity between sentence A and sentence B does not consider how sentence B is similar to the title and how many key phrases exist in sentence B. In this work, the enhancement approach to the calculation of cosine similarity will be changed. First, the similarity between sentence B and the title will be calculated using the original cosine similarity formula. Then, the key phrase score for sentence B will be calculated to account the importance of key phrases.

Keyphrase scoring is an essential step in identifying the most relevant phrases within a sentence. Several techniques can be used to extract and score keyphrases, including statistical methods like TF-IDF and YAKE, and embedding-based models such as KeyBERT.

In this work, we selected KeyBERT due to its ability to generate semantically meaningful keyphrases by leveraging contextual embeddings. Specifically, we used the AraBERT model with KeyBERT to better suit the characteristics of Arabic text.

KeyBERT calculates the relevance score of each candidate keyphrase by computing the cosine

similarity between the document (or sentence) embedding and the embedding of each candidate phrase. Higher similarity indicates higher importance.

To refine the results, we applied a thresholding step and discarded phrases with low scores. The final keyphrase score for a sentence is computed as the summation of the scores of the remaining keyphrases:

$$\text{KPS} = \sum_{i=1}^{n} \text{Score}(i) \tag{3.5}$$

Where $\text{Score}(i)$ is the KeyBERT similarity score of the i^{th} keyphrase that passed the threshold.

The similarity between any two sentences in the graph is now computed by combining their original cosine similarity, the key phrase score, and similarity to the title. The revised formula for this calculation is as follows

$$\text{sim}(A, B) = \text{CosSim}(A, B) \times (1 + \text{KPS}(B) + \text{Similarity}(B, \text{title})) \tag{3.6}$$

3.3 Ranking

This work employs two methods for ranking sentences within the graph. The first method uses weighted Degree Centrality, which evaluates sentences based on statistical features such as position, length, cue words, and numerical data. The second method adapts the PageRank algorithm, initializing scores using these statistical features to enhance the ranking process. Both approaches aim to identify the most representative and informative sentences for inclusion in the final summary.

3.3.1 Weighted Degree-Centrality

Weighted degree centrality measures the number of direct connections a sentence has with other sentences in a graph, reflecting its importance. Weights between nodes are calculated using cosine similarity. Thresholding is applied to reduce the number of edges by removing connections with weights below a specified threshold [35]. In this work, the degree of each sentence will be calculated using two different methods.

Standard Degree-Centrality. The degree of a sentence is calculated here as the total number of edges connected to it in the graph. Each edge represents a relationship between the sentence and another sentence, regardless of the weight of the connection. This method provides a simple measure of how interconnected the sentence is within the graph. The degree is calculated using the following formula:

$$\text{Degree}(S_i) = \sum_{j=1}^{N} E_{ij} \tag{3.7}$$

Where S_i is the sentence, $E_{ij} = 1$ if there is an edge between S_i and S_j, and $E_{ij} = 0$ otherwise.

N is the total number of sentences in the graph.

Weighted Degree-Centrality. The centrality of a sentence is calculated here by considering four statistical features: sentence location, sentence length, the presence of cue words, and numerical data included in the sentence. This method leverages these features to capture the structural and semantic significance of a sentence within the text. By incorporating statistical features, the approach provides a more nuanced evaluation of sentence importance, balancing both positional and content-based aspects, which can enhance the quality and relevance of the summarization process.

Sentence Location. The position of a sentence within its paragraph is an important indicator of its significance. Typically, sentences at the beginning of a paragraph are more likely to convey key information, as they often introduce the main ideas. As a result, such sentences are assigned higher importance in summarization. This feature is quantified as follows:

$$\text{location}(S_i) = \begin{cases} \frac{1}{P_i}, & \text{if } S_i \text{ is the first sentence in the paragraph} \\ \frac{1}{S_i P_i}, & \text{if } S_i \text{ is any other sentence} \end{cases} \tag{3.8}$$

Where P_i represents the paragraph index, and S_i denotes the sentence's position within the paragraph.

Sentence Length. The length of a sentence is an indicator of the amount of information it contains. Longer sentences are assumed to convey more meaningful content, making them more important for summarization. On the other hand, shorter sentences are likely to include less significant information and are given lower importance. The sentence length feature is computed using the following equation:

$$\text{Sentance Length Score} = \frac{\text{word\_count}}{\text{max\_word\_count}} \times Entropy \tag{3.9}$$

Where `word_count` refers to the number of words in the current sentence, while `max_word_count` denotes the maximum word count among all sentences in the file, excluding the title. The `Entropy` term represents the Shannon entropy of the sentence, which quantifies the diversity or unpredictability of characters within the sentence. It is defined by the equation:

$$\text{Entropy} = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{3.10}$$

Where p_i is the probability of the i-th unique character in the sentence, and n is the total number of unique characters. This entropy captures the information content of the sentence based on character distribution.

Cue Words. The importance of a sentence is determined by the presence or absence of specific cue words. Sentences containing a higher number of cue words are considered more significant and are more likely to be included in the summary. The relevance of a sentence based on cue words is calculated using the following formula:

$$\text{cueWordsScore} = \frac{\#\text{CueWordsInSentence}}{\#\text{CueWordsInParagraph}} \tag{3.11}$$

Numarical Data. Dates and numerical values often carry significant information within a text. Therefore, the presence of numerical data in a sentence increases its importance. This importance is represented using the following formula:

$$\text{numDataScore} = \frac{\#\text{NumDataInSentence}}{\#\text{NumDataInParagraph}} \tag{3.12}$$

The score of each sentence can now be determined by summing the four statistical features discussed earlier as shown in Formula 3.12. Additionally, the centrality of a sentence is computed by summing the weights of its connections with other sentences, multiplied by their respective scores, as demonstrated in Formula 3.13:

$$\text{Score}(s_i) = \text{location}(s_i) + \text{length}(s_i) + \text{cueWordsScore}(s_i) + \text{numDataScore}(s_i) \tag{3.13}$$

$$\text{Degree}(S_i) = \sum_{j \in \mathcal{N}(i)} w_{ij} \cdot \text{Score}(S_j) \tag{3.14}$$

Where $\mathcal{N}(i)$ is the set of sentences connected to sentence S_i, and w_{ij} is the weight of the edge between sentence S_i and sentence S_j. This approach fails to consider the weighted relationships between sentences and overlooks the significance of the second sentence. To address this limitation, the standard degree centrality will be replaced with weighted degree centrality, which better captures the importance of connections within the graph.

3.3.2 Hybrid Graph-Based Ranking

Standard TextRank. The classical TextRank algorithm treats each sentence as a node in a graph, where edges represent similarity between sentence pairs, and importance scores are computed iteratively. In the original formulation, the score of a sentence is updated using the following rule [39].

$$S_i^{(t)} = d + (1 - d) \sum_{j \in \text{Adj}(i)} \frac{w_{ij} \cdot S_j^{(t-1)}}{|\text{Adj}(j)|} \tag{3.15}$$

Here, $S_i^{(t)}$ is the score of sentence i at iteration t, d is the damping factor, w_{ij} is the weight from node j to node i, and $|\text{Adj}(j)|$ denotes the number of outgoing edges from node j.

Degree-Normalized TextRank. Instead of using the number of outgoing edges, we normalize the TextRank iteration update formula by the sum of the actual edge weights as follow.

$$S_i^{(t)} = d + (1-d) \sum_{j \in \text{Adj}(i)} \frac{w_{ij} \cdot S_j^{(t-1)}}{\sum_{k \in \text{Adj}(j)} w_{jk}} \tag{3.16}$$

This normalization accounts for the true strength of connections from node j, allowing more precise propagation of influence.

Symmetric Normalization TextRank. To further balance score propagation between nodes, we explore a normalization that uses the square root of the degree product of the source and target nodes as follow.

$$S_i^{(t)} = d + (1-d) \cdot \sum_{j \in \text{Adj}(i)} \left(\frac{w_{ij}}{\sqrt{\sum_{k \in \text{Adj}(i)} w_{ik} \cdot \sum_{l \in \text{Adj}(j)} w_{jl}}} \cdot S_j^{(t-1)} \right) \tag{3.17}$$

This approach prevents high-degree nodes from exerting disproportionate influence and promotes a more balanced distribution of scores across the graph.

Fusion-Based Ranking. In our final variant, the final score for each sentence is computed by combining the TextRank score from Formula 3.17 with a content-aware statistical score:

$$F(i) = \alpha \cdot \hat{S}_{\text{TextRank}}(i) + (1-\alpha) \cdot \hat{S}_{\text{Stat}}(i) \tag{3.18}$$

Here, $\hat{S}_{\text{TextRank}}(i)$ is the normalized score after ranking, and $\hat{S}_{\text{Stat}}(i)$ is a precomputed statistical score based on sentence features. The parameter $\alpha \in [0, 1]$ controls the balance between structure-based and feature-based importance.

Initialization Strategies. For each of the above formulas, we apply two different strategies to initialize sentence scores:

- **Uniform Initialization:** Every sentence begins with a score of $\frac{1}{N}$, following the standard TextRank approach.
- **Statistical Initialization:** Sentence scores are initialized using a statistical score. These scores are normalized to sum to one and offer a more content-aware starting point for the ranking process.

3.4 Sentence Selection

Once the rank for each sentence is calculated, all sentences will be arranged in ascending order of their scores. The top-ranked sentences will then be selected according to the desired summary ratio, which in our case is set to less than 50% of the original text.

Chapter 4

Experiments

Contents

4.1 Data set

In our proposed work we will use (EASC) dataset which is refers to researchers at Essex University developed the Essex Arabic Summaries Corpus (EASC), a unique collection for Arabic text summary. It includes 153 articles on various topics sourced from Arabic newspapers and Wikipedia, as well as 765 human-generated extracting summaries written using Mechanical Turk. Each article includes five reference summaries created by different individuals, ensuring a more realistic evaluation than depending on translated datasets or prior summarizers. The dataset is designed to support assessment tools such as ROUGE and AutoSummENG, which are commonly utilized for evaluating the quality of text summaries, and it supports both UTF-8 and ISO-8859-6 encodings [62].

4.2 Evaluation Measure

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a popular automated evaluation metric in text summarization research. It evaluates the quality of machine-generated summaries by comparing them to human-written reference summaries. ROUGE uses overlapping content, such as n-grams, longest common subsequences, and skip-bigrams, to assess recall, precision, and F-measures. The most used statistic is ROUGE-N, which focuses on n-gram overlap.

This method offers an objective, efficient alternative to manual evaluation, which is subjective and time-consuming [62]. ROUGE-N counts overlapping units across computer-generated and reference summaries, which can be calculated using the formula below:

$$\text{ROUGE-N} = \frac{\sum_{S \in \text{Summ}_{\text{ref}}} \sum_{\text{N-grams} \in S} \text{Count}_{\text{match}}(\text{N-gram})}{\sum_{S \in \text{Summ}_{\text{ref}}} \sum_{\text{N-grams} \in S} \text{Count}(\text{N-gram})} \tag{4.1}$$

where $\text{Count}_{\text{match}}(\text{N-gram})$ is the maximum number of the common N-grams between the set of reference summaries (Summ_{ref}) and the generated summary, and $\text{Count}(\text{N-gram})$ is the total number of N-grams in the reference summary.There are multiple versions of ROUGE-N depending on the size of the unit being considered. The most commonly used ones, as utilized in DUC 2007, are ROUGE-1 and ROUGE-2. The equations for ROUGE-1 and ROUGE-2 are provided as Equation (4.2) and Equation (4.3), respectively [62].

$$\text{ROUGE-1} = \frac{\sum_{S \in \text{Summ}_{\text{ref}}} \sum_{\text{1-grams} \in S} \text{Count}_{\text{match}}(\text{1-gram})}{\sum_{S \in \text{Summ}_{\text{ref}}} \sum_{\text{1-grams} \in S} \text{Count}(\text{1-gram})} \tag{4.2}$$

$$\text{ROUGE-2} = \frac{\sum_{S \in \text{Summ}_{\text{ref}}} \sum_{\text{2-grams} \in S} \text{Count}_{\text{match}}(\text{2-gram})}{\sum_{S \in \text{Summ}_{\text{ref}}} \sum_{\text{2-grams} \in S} \text{Count}(\text{2-gram})} \tag{4.3}$$

ROUGE-L evaluates the quality of a generated summary based on the Longest Common Subsequence (LCS) between the generated summary and reference summaries. Unlike n-gram based metrics, ROUGE-L does not require consecutive matches but instead rewards in-sequence matches that respect word order, making it more flexible and semantically meaningful.

$$\text{ROUGE-L} = \frac{\sum_{S \in \text{Summ}_{\text{ref}}} \text{LCS}_{\text{match}}(S, \text{Summ}_{\text{gen}})}{\sum_{S \in \text{Summ}_{\text{ref}}} \text{LCS}(S)} \tag{4.4}$$

In the equation, $\text{LCS}_{\text{match}}(S, \text{Summ}_{\text{gen}})$ denotes the length of the longest common subsequence between a reference summary S and the generated summary Summ_{gen}, while $\text{LCS}(S)$ is the length of the reference summary. ROUGE-L is often considered useful for capturing sentence-level structure similarity and is widely used in summarization evaluations.

4.3 Tools

In this work, all tasks related to our graph-based text summarization will be carried out using Python. The tools and libraries selected for each stage of the process are listed in Table 4.1. These tools are designed to handle various aspects of text preprocessing, sentence representation, similarity computation, graph construction and ranking, which are essential components for building the summarization system.

Task	Library	Module/Function
Stop Word Removal	nltk	nltk.corpus.stopwords
Root Stemming	nltk	nltk.stem.isri.ISRIStemmer
TF-ISF Vectorization	scikit-learn	TfidfVectorizer
Cosine Similarity Computation	scikit-learn	cosine_similarity
Graph Construction & Manipulation	NetworkX	networkx.Graph, add_node, add_edge
FastText Sentence Representation	Gensim/FastText	gensim.models.FastText, fasttext CLI
AraBERT Sentence Representation	Hugging Face Transformers	AutoTokenizer, AutoModel
Key Phrase Extraction	keybert	KeyBERT
Ranking - Degree Centrality	NetworkX	nx.degree_centrality
Ranking - PageRank	NetworkX	nx.pagerank
Numerical Data Extraction	re / SpaCy	re.findall

Table 4.1: Summary of Tasks, Libraries, and Modules/Functions

4.4 Results

4.4.1 Expirement Setup and Parameters Tuning

In this extractive graph-based Arabic text summarization approach, a preprocessing step was first applied to all documents in the dataset. Following this, several features were extracted for each sentence within each document. These features included keyphrase score, sentence length and location scores, cue word score, and numerical data score.

After preprocessing and feature extraction, a graph was constructed for each document, where sentences were treated as nodes. The edges between nodes represented the similarity between sentence pairs, which was calculated using Formula 3.6. All edge weights were subsequently normalized. Each node in the graph was represented as a vector, determined using one of three techniques: TF-ISF, FastText, or AraBERT. It was observed that omitting graph pruning yielded better results compared to applying a threshold to prune the edges.

Once the graphs were constructed, sentence ranking was performed. Experiments were conducted to rank sentences using weighted degree centrality, as described in Formula 3.14, for each of the three representation techniques. Additionally, sentence ranking was carried out using Formulas 3.16, 3.17, and 3.18. The value of alpha was set to 0.89 for TF-ISF and 0.88 for both FastText and AraBERT representations. A damping factor of 0.85 was used in experiments involving Formulas 3.16 and 3.17.

In the final step, top-ranked sentences were selected to form the summary. Where the summary ratio was set to be less than 50% of the original text length, ensuring concise and informative summaries.

4.4.2 Results Analysis

Table 4.2 presents a comparison of various graph-based extractive summarization approaches using ROUGE-1, ROUGE-2, and ROUGE-L as evaluation metrics. These approaches were tested with different sentence representations, including TF-ISF, FastText, and AraBERT, and some were also evaluated under different thresholding values.

For the Degree Centrality approach, using FastText with a threshold of 0.5 produced the best performance in this group. It achieved F1 scores of 0.6524 for ROUGE-1, 0.5928 for ROUGE-2, and 0.6458 for ROUGE-L. This indicates that combining centrality scores with FastText embeddings captures both relevant and diverse content effectively. On the other hand, using AraBERT in this setup resulted in the lowest scores among the three representation methods, suggesting a weaker compatibility with the Degree Centrality approach.

In the PageRank method, the TF-ISF representation showed the highest ROUGE-1 F1 score (0.5806), while FastText slightly outperformed it in ROUGE-L with a score of 0.5763. AraBERT again had the weakest performance in this setting, showing that its contextual embeddings might not align well with the PageRank node scoring mechanism.

TextRank results followed a similar pattern, where FastText achieved the best performance with F1 scores of 0.6074 (ROUGE-1), 0.5380 (ROUGE-2), and 0.5983 (ROUGE-L). TF-ISF had moderate performance, while AraBERT lagged behind in all three ROUGE metrics. This further supports the observation that FastText provides a better semantic representation for graph-based summarization in this context.

Among all approaches, LexRank with FastText and no thresholding stood out as the most effective configuration. It achieved an F1 score of 0.6366 for ROUGE-1, 0.5722 for ROUGE-2, and 0.6285 for ROUGE-L. These results suggest that LexRank benefits from combining structural sentence importance with FastText's semantic similarity, making it highly suitable for extractive summarization tasks.

Overall, FastText consistently outperformed both TF-ISF and AraBERT across the majority of methods. Although AraBERT is known for its strong performance in various NLP tasks, it did not perform well in this context, likely due to its contextual nature being less effective in graph structures used for extractive summarization. ROUGE-2 scores were generally lower than ROUGE-1 and ROUGE-L, which is expected as ROUGE-2 measures bigram overlaps and is a stricter metric. In conclusion, LexRank using FastText without thresholding achieved the best balance between precision and recall, making it the most promising configuration among the tested methods.

The Weighted Degree Centrality (Weighted DC) approach showed promising performance, particularly when combined with FastText embeddings and thresholding 0.5. This configuration achieved ROUGE-1, ROUGE-2, and ROUGE-L F1 scores of 0.6276, 0.5633, and 0.6203 respectively. In contrast, when using AraBERT embeddings, the performance was lower.

For Equation 3.16, both initialization strategies—uniform (1/N) and sentence score-based—yielded nearly identical results. With FastText, they both achieved strong performance, reaching F1 scores of approximately 0.6513 (ROUGE-1) and 0.644 (ROUGE-L). This demonstrates the effectiveness and consistency of the equation when paired with semantic embeddings.

Equation 3.17 followed a similar trend. Using FastText embeddings, the best configuration reached ROUGE-1, ROUGE-2, and ROUGE-L F1 scores of 0.6535, 0.5928, and 0.6462 respectively. These results confirm the stability of this approach across different initialization techniques.

The best overall performance was observed with Equation 3.18. When using TF-ISF representation and sentence score initialization, it achieved the highest F1 scores among all experiments: 0.7031 (ROUGE-1), 0.6513 (ROUGE-2), and 0.6979 (ROUGE-L). This highlights the importance of using informative initial sentence scores to guide the ranking process effectively. Even with AraBERT, the results under Equation 3.18 were competitive, showing the robustness of this approach.

Approach	**Thresholding**	**Representation**	**ROUGE-1**			**ROUGE-2**			**ROUGE-L**		
			Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Degree Centrality	0.04	TF-ISF	0.6183	0.6150	0.6096	0.5426	0.5524	0.5422	0.6093	0.6069	0.6014
	0.5	FastText	0.6165	0.7062	0.6524	0.5604	0.6414	0.5928	0.6106	0.6983	0.6458
	0.3	AraBERT	0.5627	0.5664	0.5567	0.4846	0.4885	0.4792	0.5528	0.5576	0.5477
PageRank	No Thr.	TF-ISF	0.6274	0.5595	0.5806	0.5535	0.4969	0.5135	0.6204	0.5540	0.5748
	No Thr.	FastText	0.6	0.5844	0.5835	0.529	0.5139	0.5136	0.5918	0.5775	0.5763
	No Thr.	AraBERT	0.5654	0.507	0.5231	0.4839	0.4339	0.447	0.5556	0.4997	0.515
TextRank	No Thr.	TF-ISF	0.6118	0.5437	0.5647	0.532	0.4817	0.4957	0.6047	0.5381	0.5587
	No Thr.	FastText	0.5892	0.6478	0.6074	0.5248	0.5713	0.538	0.5806	0.6373	0.5983
	No Thr.	AraBERT	0.5571	0.567	0.5542	0.478	0.4879	0.4759	0.5464	0.5572	0.5443
LexRank	0.2	TF-ISF	0.5591	0.5315	0.5367	0.4827	0.4606	0.464	0.5483	0.5219	0.5269
	No Thr.	FastText	0.6083	0.6821	0.6366	0.5471	0.6125	0.5722	0.6008	0.6726	0.6285
	No Thr.	AraBERT	0.5682	0.5306	0.5403	0.488	0.4521	0.4614	0.5581	0.522	0.5312
Wieghted DC	0.01	TF-ISF	0.5984	0.591	0.5873	0.5199	0.5274	0.5175	0.5885	0.5826	0.5786
	0.5	FastText	0.6016	0.6689	0.6276	0.5403	0.5997	0.5633	0.5948	0.6606	0.6203
	No Thr.	AraBERT	0.5495	0.5429	0.539	0.4687	0.4619	0.4588	0.5393	0.5335	0.5294
Equiasion 3.16, initialization: 1/N	No Thr.	TF-ISF	0.6224	0.6414	0.625	0.554	0.5783	0.5599	0.6146	0.6336	0.6174
	No Thr.	FastText	0.6183	0.7015	0.6513	0.5606	0.634	0.5896	0.6118	0.6929	0.644
	No Thr.	AraBERT	0.5535	0.5355	0.5367	0.472	0.4556	0.4568	0.5431	0.5265	0.5273
Equiasion 3.16, initialization: Sentence score	No Thr.	TF-ISF	0.6221	0.641	0.6247	0.5536	0.5776	0.5593	0.6143	0.6332	0.6171
	No Thr.	FastText	0.6183	0.7015	0.6513	0.5606	0.634	0.5896	0.6118	0.6929	0.644
	No Thr.	AraBERT	0.5605	0.5638	0.5544	0.481	0.485	0.4759	0.5507	0.5549	0.5453
Equation 3.17, initialization: 1/N	No Thr.	TF-ISF	0.6182	0.6339	0.6192	0.5482	0.5689	0.5525	0.6102	0.626	0.6115
	No Thr.	FastText	0.6209	0.7031	0.6535	0.5637	0.637	0.5928	0.6144	0.6944	0.6462
	No Thr.	AraBERT	0.5505	0.5405	0.5379	0.47	0.4608	0.4586	0.54	0.5311	0.5282
Equation 3.17, initialization: Sentence score	No Thr.	TF-ISF	0.6179	0.6336	0.6189	0.5477	0.5682	0.5519	0.6099	0.6257	0.6112
	No Thr.	FastText	0.6209	0.7031	0.6535	0.5637	0.637	0.5928	0.6144	0.6944	0.6462
	No Thr.	AraBERT	0.5505	0.5405	0.5379	0.47	0.4608	0.4586	0.54	0.5311	0.5282
Equation 3.18, initialization: 1/N	No Thr.	TF-ISF	0.6681	0.7551	0.703	0.6179	0.7018	0.6515	0.6632	0.7492	0.6979
	No Thr.	FastText	0.6527	0.7418	0.6879	0.6019	0.6832	0.634	0.6471	0.7344	0.6817
	No Thr.	AraBERT	0.6559	0.7444	0.6908	0.6048	0.6856	0.6366	0.6504	0.7374	0.6849
Equation 3.18, initialization: Sentence score	No Thr.	TF-ISF	0.668	0.7559	0.7031	0.6177	0.7019	0.6513	0.6629	0.75	0.6979
	No Thr.	FastText	0.6527	0.7418	0.6879	0.6019	0.6832	0.634	0.6471	0.7344	0.6817
	No Thr.	AraBERT	0.657	0.7448	0.6917	0.6061	0.6866	0.6378	0.6516	0.7377	0.6857

Table 4.2: ROUGE scores for different summarization approaches and representations

Table 4.3 compares the output summaries generated by each method (PageRank, TextRank, and Best of 3.16–3.18) for File 139 from the dataset. To ensure parity with the ground truth (golden summary), all methods were constrained to select exactly 2 sentences, mirroring the length of the reference summary.

Input Text	الملاكمة ، الملقبة أيضاً برياضة الملوك ، هي رياضة يهاجم فيها اثنان من الرياضيين ذوي الوزن المماثل بعضهما البعض بقبضاتهم في سلسلة فترات تتراوح من 1 إلى 3 دقائق تسمى "الجولات" . في كل من الانقسامات الأوليمبية والمحترفة ، المقاتلون الذين يدعون الملاكمين أو المقاتلين يتفادون لكمات معارضهم بينما يحاولون تحقيق لكمات بأنفسهم . النقاط ممنوحة للضربات الصلبة النظيفة في المنطقة القانونية على جبهة جسم المعارض فوق الخصر ، والضربات إلى الرأس والجذع يعتبران أثمن . إن المقاتل الحاصل على أكثر النقاط بعد العدد المحدد من الجولات يعلن فائزاً . النصر يمكن أن ينجز أيضاً إذا سقط المعارض وأصبح غير قادر على النهوض قبل أن يحسب الحكم إلى عشرة وتسمى الضربة القاضية ، أو أو إذا المعارض كان مصاباً جداً ولا يمكنه أن يستمر .
Ground Truth	الملاكمة، الملقبة أيضاً برياضة الملوك، هي رياضة يهاجم فيها اثنان من الرياضيين ذوي الوزن المماثل بعضهما البعض بقبضاتهم في سلسلة فترات تتراوح من 1 إلى 3 دقائق تسمى "الجولات". النصر يمكن أن يُنجز أيضاً إذا سقط المعارض وأصبح غير قادر على النهوض قبل أن يحسب الحكم إلى عشرة وتسمى الضربة القاضية، أو إذا كان المعارض مصاباً جداً ولا يمكنه أن يستمر.
PageRank	إن المقاتل الحاصل على أكثر النقاط بعد العدد المحدد من الجولات يعلن فائزاً. النصر يمكن أن ينجز أيضاً إذا سقط المعارض وأصبح غير قادر على النهوض قبل أن يحسب الحكم إلى عشرة وتسمى الضربة القاضية ، أو أو إذا المعارض كان مصاباً جداً ولا يمكنه أن يستمر
TextRank	إن المقاتل الحاصل على أكثر النقاط بعد العدد المحدد من الجولات يعلن فائزاً. النصر يمكن أن ينجز أيضاً إذا سقط المعارض وأصبح غير قادر على النهوض قبل أن يحسب الحكم إلى عشرة وتسمى الضربة القاضية ، أو أو إذا المعارض كان مصاباً جداً ولا يمكنه أن يستمر
Best of 3.15	الملاكمة ، الملقبة أيضاً برياضة الملوك ، هي رياضة يهاجم فيها اثنان من الرياضيين ذوي الوزن المماثل بعضهما البعض بقبضاتهم في سلسلة فترات تتراوح من 1 إلى 3 دقائق تسمى "الجولات". النصر يمكن أن ينجز أيضاً إذا سقط المعارض وأصبح غير قادر على النهوض قبل أن يحسب الحكم إلى عشرة وتسمى الضربة القاضية ، أو أو إذا المعارض كان مصاباً جداً ولا يمكنه أن يستمر
Best of 3.16	الملاكمة ، الملقبة أيضاً برياضة الملوك ، هي رياضة يهاجم فيها اثنان من الرياضيين ذوي الوزن المماثل بعضهما البعض بقبضاتهم في سلسلة فترات تتراوح من 1 إلى 3 دقائق تسمى "الجولات". النصر يمكن أن ينجز أيضاً إذا سقط المعارض وأصبح غير قادر على النهوض قبل أن يحسب الحكم إلى عشرة وتسمى الضربة القاضية ، أو أو إذا المعارض كان مصاباً جداً ولا يمكنه أن يستمر
Best of 3.17	الملاكمة ، الملقبة أيضاً برياضة الملوك ، هي رياضة يهاجم فيها اثنان من الرياضيين ذوي الوزن المماثل بعضهما البعض بقبضاتهم في سلسلة فترات تتراوح من 1 إلى 3 دقائق تسمى "الجولات". النصر يمكن أن ينجز أيضاً إذا سقط المعارض وأصبح غير قادر على النهوض قبل أن يحسب الحكم إلى عشرة وتسمى الضربة القاضية ، أو أو إذا المعارض كان مصاباً جداً ولا يمكنه أن يستمر

Table 4.3: Output Summaries for File 139

4.4.3 Results Discussion

From the experimental results presented in Table 4.2, we observed that although AraBERT is a state-of-the-art contextual model, it did not consistently outperform simpler representations like TF-ISF and FastText. This outcome can be attributed to the nature of extractive summarization

and the evaluation metric used—namely, ROUGE—which emphasizes lexical overlap over semantic depth.

Graph-based summarization relies heavily on explicit similarity measures that reward surface-level lexical similarity. The ROUGE metric, which calculates n-gram co-occurrence with reference summaries, inherently favors models that maintain exact phrasing. In this context, TF-ISF proves particularly effective. By generating sparse, high-dimensional vectors, it enables cosine similarity to capture term overlap, thereby highlighting sentences containing key terms that are likely to appear in human-written summaries.

Conversely, AraBERT's ability to understand deep contextual and semantic relationships does not always yield higher ROUGE scores. Several factors contribute to this:

- **Embedding Aggregation:** Averaging token-level embeddings into a single sentence vector can obscure the importance of key terms, weakening the model's effectiveness in graph-based settings.
- **Semantic vs. Lexical Similarity:** AraBERT may select semantically relevant sentences that are lexically different from the reference summary. While semantically correct, such choices are penalized by ROUGE for lacking exact n-gram matches.
- **Morphological Complexity:** AraBERT's subword tokenization can fragment key lexical units in morphologically rich languages like Arabic. In contrast, TF-ISF, especially when combined with normalization or stemming, preserves the lexical roots more effectively.

In summary, while AraBERT offers a sophisticated understanding of language, its performance is constrained by the evaluation framework, which prioritizes lexical fidelity. TF-ISF, with its direct emphasis on term salience, aligns more closely with the ROUGE metric, leading to better performance in graph-based extractive summarization.

In contrast, FastText provided a robust middle ground between contextual richness and structural simplicity. It consistently improved sentence similarity calculations, leading to higher ROUGE scores across nearly all methods. Its ability to represent subword information is especially valuable in Arabic, where morphology plays a major role in word formation.

Among the proposed methods, Equation 3.18 with sentence score initialization emerged as the most effective. This suggests that integrating initial knowledge about sentence importance—through statistical cues like length, position, and keyphrases—can guide the ranking process more efficiently. While this initialization did not always boost final ROUGE scores compared to uniform initialization, it consistently reduced the number of iterations required to converge, as evidenced by convergence plots for large documents in Figure 4.1. Notably, our model converged approximately $5.15\times$ faster than PageRank and $4.85\times$ faster than TextRank, highlighting its advantage in computational efficiency. This trade-off between computational efficiency and output quality is particularly relevant in large-scale applications.

Lastly, while LexRank with FastText was the best among traditional methods, all variants of our proposed ranking equations outperformed it. This demonstrates the effectiveness of our

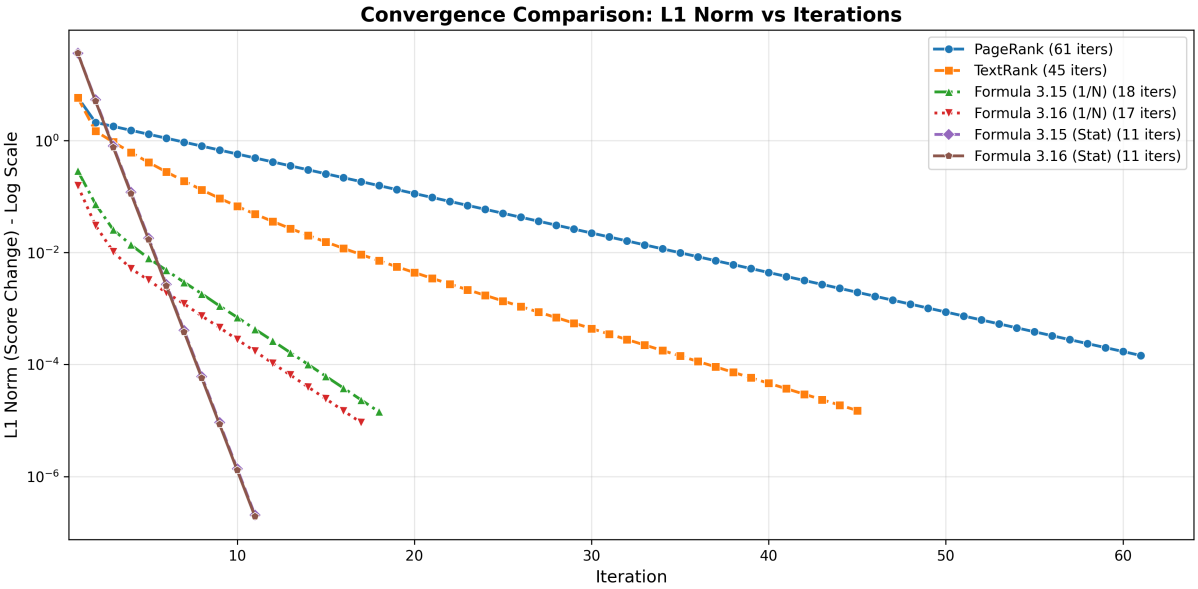

Figure 4.1: L1 norm convergence (log scale) showing faster convergence of proposed formulas (3.16, 3.17) compared to PageRank and TextRank.

enhancements—particularly the use of weighted edges and initialization strategies—in capturing both sentence relevance and diversity in summaries.

4.5 Comparison With Related Work

Table 4.4 presents a comparison between our proposed summarization system and three existing systems using ROUGE-1 evaluation metrics, which is widely used in most of the related work we reviewed. As shown, the Graph-Based Summarizer and EmbedRank Summarizer perform reasonably well, achieving F1 scores of 0.617 and 0.6521 respectively. The Aggregate Similarity Summarizer shows the lowest performance, with an F1 score of only 0.461, which indicates that it may not capture key information as effectively.

In contrast, **our proposed system outperforms all baselines**, achieving the highest scores across all three ROUGE-1 metrics. It reaches a **precision of 0.668**, a **recall of 0.7559**, and an **F1 score of 0.7031**, demonstrating its ability to both identify important content and express it accurately. This improvement highlights the effectiveness of combining graph-based techniques with semantic and statistical features in our approach.

System	Approach	Precision	Recall	F1
Graph-Based Summarizer [63]	Statistical + Structural: Constructs a graph using TF-IDF and cosine similarity. Utilizes sentence-level features such as title presence, position, and length. Only triangle-connected sentences are retained.	0.601	0.633	0.617
EmbedRank Summarizer [64]	Semantic + Ranking: Represents sentences using Word2Vec embeddings. Builds a similarity graph and ranks sentences with the PageRank algorithm.	0.6525	0.6569	0.6521
Aggregate Similarity Summarizer [65]	Semantic + Statistical: Segments text into Elementary Discourse Units (EDUs). Applies rhetorical structure analysis and sentence compression. Sentences are scored based on similarity to the title, length, and position.	0.471	0.453	0.461
Our Proposed Work	Graph-based + Semantic & Statistical Features: Builds a similarity graph using semantic signals (title and keyphrase similarity) and enhances TextRank with four statistical features: position, location, cue words, and numerical data.	**0.668**	**0.7559**	**0.7031**

Table 4.4: Comparison of summarization systems using ROUGE-1 scores

4.6 System Implementation

To demonstrate the effectiveness of our Arabic text summarization approach, we developed a complete web-based system that allows users to input or upload Arabic texts and receive extractive summaries. The backend was implemented using the Flask framework and employs the TF-ISF sentence representation combined with our proposed ranking formula Formula 3.18, which achieved the best ROUGE evaluation scores during experimentation. Once the user submits a request, the backend processes the input and returns a list of sentences ranked by importance.

The frontend is built using React framework. Users can either paste Arabic text into a text area or upload a `.txt` file. They may also specify a desired summary ratio. The browser sends the input to the backend, which computes sentence rankings and returns them. On the client side, changing the summary ratio instantly updates the displayed summary without further backend calls. For deployment, the backend runs on Railway [66], and the frontend is hosted on Netlify [67].

Figure 4.2 shows the empty state of the application. The user is presented with:

- A large text area for pasting Arabic text.
- An "Upload `.txt`" button for file input.
- A summary ratio control.

- A “Summarize” button to submit the text.

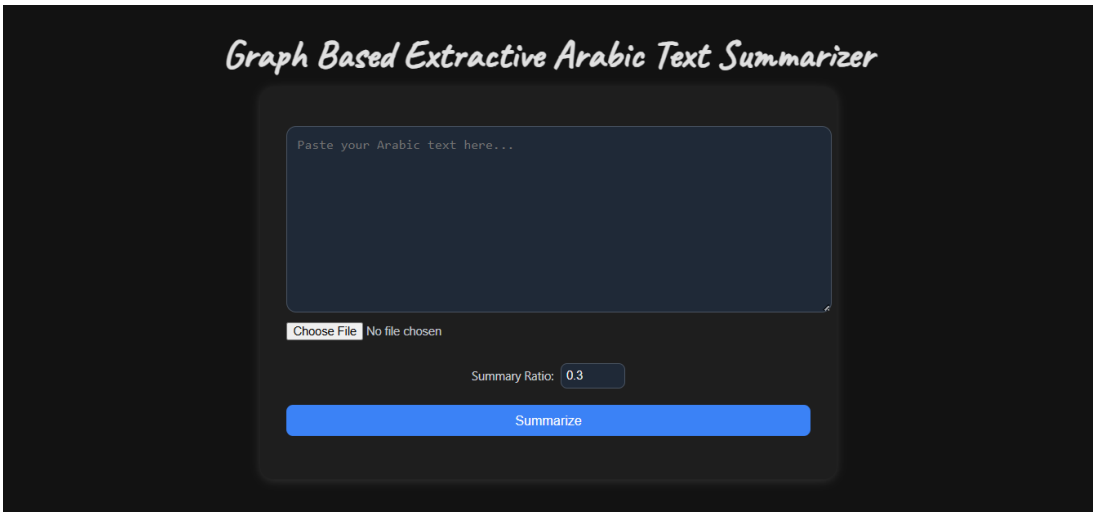

Figure 4.2: Initial system interface before any text is entered or file uploaded.

In Figure 4.3, the user has entered a block of Arabic text and chosen a summary ratio. At this point, clicking “Summarize” will send the request to the backend.

Figure 4.3: Interface after pasting text (or uploading a file) and setting the summary ratio.

Figure 4.4 displays the top-ranked sentences as the extractive summary. Notice that:

- The summary appears immediately below the controls.
- Adjusting the ratio slider dynamically recalculates and redisplays the summary on the client side.

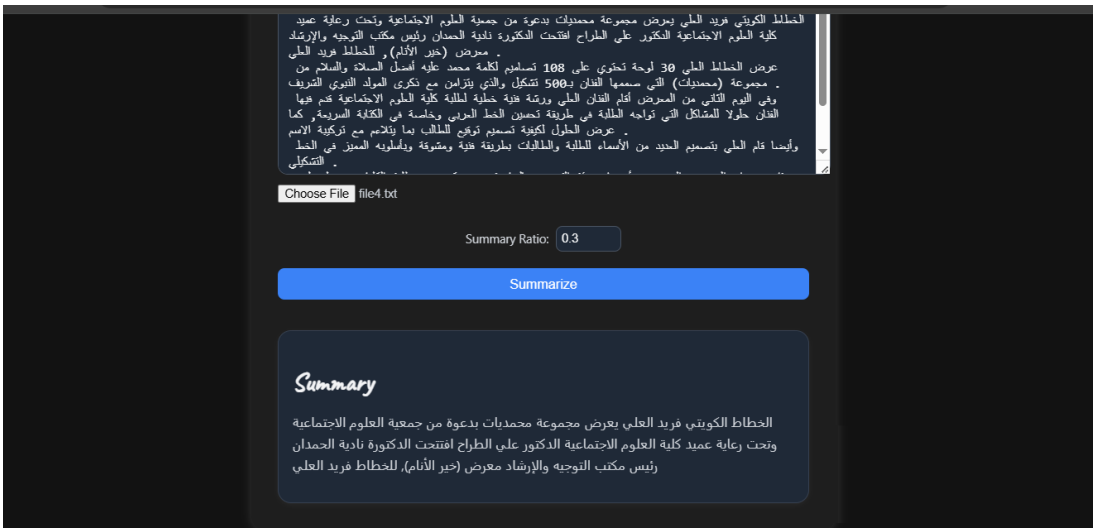

Figure 4.4: Displayed extractive summary after processing.

Chapter 5

Conclusion and Future Work

This project presented a graph-based extractive summarization system tailored for Arabic text, addressing both the semantic complexity and structural characteristics of the language. Our work combined statistical and semantic insights to improve sentence ranking in the summarization process. Through enhanced edge weight calculations—incorporating cosine similarity with additional features like keyphrase score and title similarity—we strengthened the graph's ability to capture meaningful sentence relationships. Furthermore, we introduced a novel initialization strategy for the ranking algorithm based on statistical sentence features, such as length, position, cue words, and numerical data. This approach improved convergence speed significantly while maintaining competitive ROUGE scores, demonstrating a practical balance between performance and computational efficiency.

Our experimental evaluation, conducted on the Essex Arabic Summaries Corpus, confirmed the effectiveness of the proposed methods. FastText emerged as a particularly useful sentence embedding technique due to its ability to model subword information in morphologically rich languages like Arabic. Although AraBERT is a powerful contextual model, it did not consistently outperform simpler methods in this extractive, graph-based setup. The results also validated the robustness of our enhanced ranking equations across various embeddings and initialization schemes.

Looking ahead, there are several promising directions for future development. One major enhancement involves expanding the system's input capabilities to support additional document formats beyond plain text, such as PDF, Word (.docx), and other common file types. This would require integrating document parsing and preprocessing pipelines that can handle encoding issues, extract meaningful content, and ignore non-textual elements such as headers or footers. Moreover, future work can explore integrating abstractive techniques alongside our extractive framework to generate more coherent and fluent summaries. Lastly, incorporating user-specific preferences or summary customization features—such as preferred summary length or focus keywords—can further improve the system's usability and adaptability in real-world applications.

Bibliography

[1] P. State, "How many journal articles have been published?." https://publishingstate.com/how-many-journal-articles-have-been-published/2023/, 2023. Accessed: January 2, 2025.

[2] WordsRated, "Number of academic papers published per year." https://wordsrated.com/number-of-academic-papers-published-per-year/, 2023. Accessed: January 2, 2025.

[3] R. I. for the Study of Journalism, "Digital news report 2023. available at: https://reutersinstitute.politics.ox.ac.uk/digital-news-report/2023," 2023. Accessed: January 2, 2025.

[4] Supriyono, A. P. Wibawa, Suyono, and F. Kurniawan, "A survey of text summarization: Techniques, evaluation, and challenges," *Journal of Emerging Technologies in Web Intelligence*, vol. 15, no. 4, pp. 258–268, 2023.

[5] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed, "Automatic text summarization: A comprehensive survey," *Expert Systems with Applications*, vol. 165, p. 113679, 2021.

[6] H. Zhang, P. S. Yu, and J. Zhang, "A systematic survey of text summarization: From statistical methods to large language models," *ACM Computing Surveys*, 2023.

[7] Y. A. Al-Khassawneh and E. S. Hanandeh, "Extractive arabic text summarization-graph-based approach," *Electronics*, vol. 12, no. 2, p. 437, 2023.

[8] K.-E. C. Yao-Ting Sung and T.-C. Liu, "The effects of integrating mobile devices with teaching and learning on students' learning performance: A meta-analysis and research synthesis," *Computers & Education*, vol. 94, pp. 252–275, 2016.

[9] V. Gupta and G. S. Lehal, "A survey of text summarization extractive techniques," *Journal of Emerging Technologies in Web Intelligence*, vol. 2, pp. 258–268, 2010.

[10] M. I. Hashem, "Improvement of email summarization using statistical based method," *International Journal of Computer Science and Mobile Computing*, vol. 3, pp. 382–388, 2014.

[11] S. Pathak and T. B. Dhamala, "Topic modeling based extractive text summarization," *arXiv preprint arXiv:2106.15313*, 2021. Section on Topic-Based Method, page 2 - 3.

[12] S. Topics, "Topic modeling," 2024.

[13] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed, "Automatic text summarization: A comprehensive survey," *Expert Systems with Applications*, vol. To appear, 2020. Topic-Based Method discussed on page 9.

[14] S. Liu, M. X. Zhou, S. Pan, W. Qian, W. Cai, and X. Lian, "Interactive, topic-based visual text summarization and analysis," in *Proceedings of the 2009 ACM Conference on Computer Supported Cooperative Work*, (Beijing, China; Hawthorne, NY, USA), pp. 501–510, ACM, 2009.

[15] M. G. Ozsoy, F. N. Alpaslan, and I. Cicekli, "Text summarization using latent semantic analysis," *Journal of Information Science*, vol. 36, no. 4, pp. 410–425, 2010.

[16] J. Steinberger and K. Ježek, "Using latent semantic analysis in text summarization and summary evaluation," *Computational Linguistics*, vol. 30, no. 4, pp. 581–593, 2004.

[17] T. D. Science, "Document summarization using latent semantic indexing," 2020. Accessed: 2024-10-14.

[18] S. Mandal and G. K. Singh, "Lsa based text summarization," *Journal of Information Science*, vol. 36, no. 4, pp. 410–425, 2010.

[19] M. I. Yusuf, M. Al-Fityan, and I. A. Latif, "Multi-document summarization using semantic role labeling and semantic graph for indonesian news article," in *2021 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, pp. 1–5, IEEE, 2021.

[20] Z. Amir, B. Kahar, and A. Zainal, "Srl-esa-textsum: A text summarization approach based on semantic role labeling and explicit semantic analysis," Tech. Rep. NBN:fife2020042322174, University of Oulu, 2020.

[21] M. Mohamed and M. Oussalah, "Srl-esa-textsum: A text summarization approach based on semantic role labeling and explicit semantic analysis," *Oulu University Repository*, 2020.

[22] H. K. Gianey and R. Choudhary, "Comprehensive review on supervised machine learning algorithms," in *2017 International Conference on Machine Learning and Data Science (MLDS)*, pp. 37–43, IEEE, 2017.

[23] A. G. Nisha Yadav, Rajeev Kumar and A. U. Khan, "Extraction-based text summarization and sentiment analysis of online reviews using hybrid classification method," *2019 11th International Conference on Wireless and Optical Communications Networks (WOCN)*, pp. 1–5, 2019.

[24] R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence rnns and beyond," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL 2016)*, pp. 280–290, Association for Computational Linguistics, 2016.

[25] J. Cheng and M. Lapata, "Neural summarization by extracting sentences and words," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 484–494, Association for Computational Linguistics, 2016.

[26] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, (Barcelona, Spain), pp. 404–411, Association for Computational Linguistics, July 2004.

[27] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[28] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, (Singapore), pp. 451–458, Association for Computational Linguistics, 2009.

[29] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, vol. 27, pp. 3104–3112, 2014.

[30] R. Paulus, C. Xiong, and R. Socher, "A deep reinforcement learning model for abstractive summarization," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, 2017.

[31] M. H. Wahab, N. A. W. A. Hamid, N. Ali, S. K. Subramaniam, R. Latip, and M. Othman, "A review on optimization-based automatic text summarization approach," *IEEE Access*, vol. 12, pp. 234567–234579, 2024.

[32] S. Aslam, F. Sarwar, M. N. Baig, N. Iqbal, and A. S. Khattak, "A review of nature-inspired algorithms on single-objective optimization problems from 2019 to 2023," *Artificial Intelligence Review*, vol. 56, pp. 5433–5487, 2023.

[33] Y. Xie, X. Chen, C. Zhang, and Y. Zhang, "Automatic multi-documents text summarization by a large-scale sparse multi-objective optimization algorithm," *Journal of Intelligent Systems*, vol. 32, no. 1, pp. 1024–1035, 2023.

[34] A. A. Bichi, R. Samsudin, R. Hassan, L. R. Hasan, and A. Ado Rogo, "Graph-based extractive text summarization method for hausa text," *PLOS ONE*, vol. 18, no. 5, p. e0285376, 2023.

[35] G. A. Gaphari, F. M. Ba-Alwi, and A. Moharram, "Text summarization using centrality concept," *International Journal of Computer Applications*, 2013.

[36] A. A. Bichi and Others, "Graph-based extractive text summarization method for hausa text," *PLoS ONE*, vol. 18, no. 5, 2023.

[37] S. Hariharan and R. Srinivasan, "Enhancements to graph-based methods for single document summarization," *International Journal of Engineering and Technology*, vol. 2, no. 1, pp. 45–56, 2010.

[38] S. Brin and L. Page, "The pagerank citation ranking: Bringing order to the web," *Technical Report*, 1998.

[39] R. Mihalcea and P. Tarau, "Textrank: Bringing order into texts," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, (Barcelona, Spain), pp. 404–411, Association for Computational Linguistics, 2004.

[40] R. Elbarougy, G. Behery, and A. El Khatib, "Extractive arabic text summarization using modified pagerank algorithm," *International Journal of Computational Linguistics and Natural Language Processing*, 2019.

[41] A. Bichi, R. Samsudin, R. Hassan, L. Hasan, and R. A. Ado, "Graph-based extractive text summarization method for hausa text," *PLoS ONE*, vol. 18, no. 5, p. e0285376, 2023.

[42] V. Gulati, D. Kumar, D. E. Popescu, and J. D. Hemanth, "Extractive article summarization using integrated textrank and bm25+ algorithm," *Electronics*, vol. 12, no. 2, p. 372, 2023.

[43] M. S. Uddin and J. C. Bansal, "A comparative analysis of textrank and lexrank algorithms using text summarization," in *Proceedings of the International Joint Conference on Advances in Computational Intelligence (IJCACI 2023)*, Algorithms for Intelligent Systems, Springer, 2023.

[44] S. Deo and D. Banik, "Text summarization using textrank and lexrank through latent semantic analysis," in *Proceedings of the 2024 Conference on Artificial Intelligence and Data Science*, (India), IEEE, 2024. IEEE Xplore.

[45] N. Alzuhair and A. Al-Dhelaan, "An approach for combining multiple weighting schemes and ranking methods in graph-based multi-document summarization," in *2019 IEEE 7th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 135–142, IEEE, 2019.

[46] M. Mosbah and S. Qarash, "Arabic text classification using tf-idf and preprocessing techniques," *Journal of Computer Science and Technology*, vol. 18, no. 2, pp. 34–45, 2023.

[47] A. Alghamdi, R. Alharthi, and M. Alamri, "Feature extraction and normalization techniques for arabic text classification," *International Journal of Computer Science*, vol. 12, no. 3, pp. 55–62, 2017.

[48] M. Attia, "Arabic tokenization system," in *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources* (V. Cavalli-Sforza and I. Zitouni, eds.), (Prague, Czech Republic), pp. 65–72, Association for Computational Linguistics, June 2007.

[49] A. Ayedh, G. Tan, K. Alwesabi, and H. Rajeh, "The effect of preprocessing on arabic document categorization," *Algorithms*, vol. 9, no. 2, p. 27, 2016.

[50] R. A. S. G. Kanan, J. Jaam, and E. Hailat, "Stop-word removal algorithm for arabic language," in *International Conference on Information and Communication Technologies: From Theory to Applications*, IEEE, 2004.

[51] K. Darwish, "Building a shallow arabic morphological analyzer in one day," in *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, pp. 1–8, Association for Computational Linguistics, 2002.

[52] L. S. Larkey and M. E. Connell, "Light stemming for arabic information retrieval," *Arabic Computational Morphology*, pp. 221–243, 2002.

[53] A. Khurshid, M. K. Bashir, and G. Rasool, "Comparison of stemming approaches for sentiment analysis in arabic," in *2019 International Conference on Frontiers of Information Technology (FIT)*, pp. 271–276, IEEE, 2019.

[54] A. M. A. Nada, E. Alajrami, A. A. Al-Saqqa, and S. S. Abu-Naser, "Arabic text summarization using arabert model using extractive text summarization approach," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 18, no. 4, pp. 1–10, 2020.

[55] Hyperskill, "Pros and cons of tf-idf," 2024. Accessed: 2024-12-23.

[56] V. Rao *et al.*, "Bert and fasttext embeddings for automatic detection of toxic speech," in *2020 IEEE 20th International Conference on Data Mining (ICDM)*, pp. 1123–1132, IEEE, 2020.

[57] L. Afuan and N. Hidayat, "Sentiment analysis of the kampusmerdeka program on twitter using support vector machine and a feature extraction comparison: Tf-idf vs. fasttext," *Journal of Advanced Data Science*, 2024. Received: July 19, 2024; Revised: August 13, 2024; Accepted: September 11, 2024; Available online: October 15, 2024.

[58] A. Roc, "Understanding fasttext: An embedding to look forward to," 2024. Accessed: 2024-12-23.

[59] W. Antoun, F. Baly, and H. Hajj, "Arabert: Transformer-based model for arabic language understanding," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2020.

[60] A. M. Abu Nada, E. Alajrami, A. A. Al-Saqqa, and S. S. Abu-Naser, "Arabic text summarization using arabert model using extractive text summarization approach," *Journal of Arabic Computational Linguistics*, vol. 12, no. 1, pp. 45–60, 2020.

[61] Y. Januzaj and A. Luma, "Cosine similarity – a computing approach to match similarity between higher education programs and job market demands based on maximum number of common words," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 17, no. 12, pp. 123–137.

[62] A. Qaroush, I. Abu Farha, W. Ghanem, M. Washaha, and E. Maali, "An efficient single document arabic text summarization using a combination of statistical and semantic features," *Journal of King Saud University – Computer and Information Sciences*, 2024. Accessed: 2024-12-26.

[63] Y. A. Al-Khassawneh and E. S. Hanandeh, "Extractive arabic text summarization—graph-based approach," *Electronics*, vol. 12, no. 2, p. 437, 2023.

[64] G. Alselwi and T. Taşcı, "Extractive arabic text summarization using pagerank and word embedding," *Arabian Journal for Science and Engineering*, vol. 49, no. 9, pp. 13115–13130, 2024.

[65] Q. Al-Radaideh and M. Afif, "Arabic text summarization using aggregate similarity," in *Proceedings of the International Arab Conference on Information Technology (ACIT2009)*, (Yemen), December 2009.

[66] Railway, "Railway - infrastructure for developers." `https://railway.app`, 2024. Accessed: 2025-06-27.

[67] Netlify, "Netlify - the platform for frontend developers." `https://www.netlify.com`, 2024. Accessed: 2025-06-27.