SUMMARIZING ARABIC ARTICLES USING LARGE LANGUAGE MODELS

Bader Alshemaimri, Ibrahim Alrayes, Turki Alothman, Fahad Almalik, Mohammed Almotlaq

Department of Software Engineering, King Saud University, Saudi Arabia

Abstract

This paper explores abstractive and extractive Arabic text summarization using AI, employing fine-tuning and unsupervised machine learning techniques. We investigate the adaptation of pre-trained language models such as AraT5 through fine-tuning. Additionally, we explore unsupervised methods leveraging unlabeled Arabic text for generating concise and coherent summaries by utilizing different vectorizers and algorithms. The proposed models are rigorously evaluated using text-centric metrics like ROUGE [1]. The research contributes to the development of robust Arabic summarization systems, offering culturally sensitive and contextually aware solutions. By bridging the gap between advanced AI techniques and Arabic language processing, this work fosters scalable and effective summarization in the Arabic domain.

Keywords

Arabic Text Summarization, Abstractive Summarization, Extractive Summarization, Natural Language Processing (NLP), Fine-Tuning Language Models

1. INTRODUCTION

In recent years, the explosive growth of digital content in Arabic has led to an overwhelming abundance of information across various domains. As a result, the need for efficient and effective methods of summarizing Arabic text has become increasingly crucial for facilitating comprehension and knowledge extraction. Automatic text summarization, a subfield of natural language processing (NLP), aims to generate concise and coherent summaries that capture the main ideas and salient information present in a given document. Within this realm, two predominant approaches have emerged: abstractive and extractive summarization.

Abstractive summarization involves generating summaries by paraphrasing and rephrasing the original text in a more concise form, often with the ability to generate novel sentences. Extractive summarization, on the other hand, involves selecting and condensing existing sentences or phrases from the source text to create a summary. Both approaches have their strengths and challenges, making it imperative to explore how AI-powered techniques, such as fine-tuning and unsupervised machine learning, can enhance the quality and accuracy of Arabic text summarization.

The aim of this research is to delve into Arabic text summarization, addressing the critical need for advanced methods in handling the language's digital content surge. By navigating the complexities of Arabic text preprocessing and exploring both extractive and abstractive summarization, this study seeks to set new benchmarks in the field. It leverages fine-tuning and unsupervised learning to push the boundaries of what's achievable in summarizing Arabic texts, contributing to this underrepresented language area in NLP.

1.1. Background Context

Automatic text summarization is a crucial task in NLP, particularly in the context of the Arabic language, which faces unique challenges due to linguistic complexities and limited labeled data. To address these challenges, researchers have explored advanced AI techniques such as fine-tuning and unsupervised machine learning.

In the pursuit of abstractive summarization for Arabic text, the present study incorporates fine-tuning the AraT5 base model [2]. AraT5, a pre-trained language model for Arabic text, is adapted to the text generation task.

Fine-tuning AraT5 or any LLM on a specific task will leverage its pre-existing knowledge and tailoring it to the intricacies of the Arabic language.

Additionally, for extractive summarization, the study explores the utilization of K-means clustering and Latent Semantic Analysis (LSA), which are unsupervised machine learning techniques. These techniques are applied to identify and condense the most relevant sentences from the source text, forming an extractive summary. These unsupervised approaches do not require labeled data, making them particularly valuable for summarizing vast volumes of Arabic text without the need for manual annotations. Moreover, the study will explore and compare different vectorizers.

1.2. Research Objectives

1. The paper aims to investigate both abstractive and extractive approaches to Arabic text summarization using AI techniques
2. The paper focuses on adapting pre-trained language models, such as AraT5, through fine-tuning to mitigate the challenge of limited labeled data for Arabic summarization 3. The study explores unsupervised techniques for generating concise and coherent summaries from unlabeled Arabic text, contributing to the development of scalable summarization systems
3. Evaluate and compare the proposed models using language-centric metrics like ROUGE, providing a comprehensive assessment of their effectiveness in generating high-quality summaries

2. LITERATURE REVIEW

This section compares the research on Arabic texts that have been done on extractive, abstractive, and hybrid summarization.

In the study conducted by [3], they introduced a novel approach to Arabic text summarization by employing multilingual BERT [4]. Their primary aim was to demonstrate the effectiveness of multilingual BERT in the context of a language like Arabic, which lacks abundant linguistic resources. Typically, a BERT model comprises an encoder and a decoder to process and generate content. The encoder utilizes various linguistic symbols and interval segmentation embedding to distinguish between related sentences. However, it's worth noting that the standard BERT model is designed for the English language. To adapt it for Arabic, the researchers utilized a multilingual BERT model that had been pre-trained in multiple languages. For their experiments and model evaluation, they employed the KALIMAT dataset, a versatile Arabic corpus containing 20,291 articles. The results of their tests on this dataset indicated an accuracy rate of 12.21% in ROUGE-1.

Another proposed model was a hybrid approach to single-document Arabic text summarization, with a focus on ASDKGA [5]. Their method integrated statistical techniques, genetic algorithms, and domain expertise, specifically tailored for political texts. The researchers utilized two datasets, the Kalimat corpus [6] and the EASC [7], and assessed the performance of their proposed model using the Rouge metric.

[8] introduced an abstractive approach for summarizing single documents in Arabic. They adapted and implemented the latest techniques and mechanisms from English to Arabic using a multi-layered LSTM transformer (Long Short-Term Memory). To emphasize the identification of critical sections in the original text, they employed an encoder with a 128-dimensional embedding layer and 256 hidden states in both directions. Their decoder featured 512 hidden states. For precision, they incorporated a copy mechanism that selected terms from the original text for inclusion in the generated summary. They compared the results of a model with only an attention mechanism to one with both attention and copy mechanisms, showing that accuracy improved with the latter. Additionally, they used Dagrad as an optimizer.

In their research conducted, [9] utilized a substantial dataset of 79,965 texts from diverse sources to develop an innovative model. This model comprised three LSTM encoder layers and one decoder layer. They implemented a three-stage process for the encoder, involving word embeddings. The first layer contained the embedding of the input text, followed by input text keywords in the second layer, and input text named entities in the third layer. In contrast, the decoder layer utilized word embeddings for the summary words. Their model's performance evaluation considered both quantitative and qualitative aspects, where they quantitatively assessed it using ROUGE.

[10]introduced a novel theory in the realm of extractive summarization, which entails evaluating sentences by considering both statistical and semantic features. Their research involved a comparative analysis of two distinct summarization methods to gauge the effectiveness of the proposed enhancements. The first method is a scoring-based approach that takes into consideration a wide range of attributes, including the frequency of strong words and their placement within phrases, to derive an overall score. The second method involves supervised learning and addresses a binary classification problem. To assess their results, they utilized five standard classifiers, namely Naive Bayes, Support Vector Machine (with an RBF kernel), Two-Layer Neural Network, and Random Forest, employing the WEKA program with 100 random trees.

2.1. Existing Approaches to Fine-Tuning LLM Models

1. *Fine Tuning* is a process that leverages the knowledge acquired by a pre-trained network and adapts it to a specific downstream task. During fine-tuning, the weights and parameters of the pre-trained network are used as a starting point for training on the new task, allowing the model to specialize and excel in the target domain. By building upon the previously learned features, the fine-tuned model can achieve better performance and faster convergence compared to training from scratch.
2. *Parameter-Efficient Fine-Tuning (PEFT)* is an advanced technique in the field of transfer learning that aims to improve the efficiency of fine-tuning pre-trained language models [11]. It addresses the issue of high computational and memory requirements typically

associated with traditional fine-tuning approaches, especially when dealing with large language models like BERT or GPT-3.

3. *Few-Shot Learning* aims to fine-tune LLM models on tasks with limited data by using only a small number of task-specific examples. Techniques such as meta-learning and episodic training are employed to enable adaptation to new tasks with minimal examples [12] .

2.2. Gaps and Limitations in the Existing Literature

The literature review reveals certain gaps and limitations within the existing research landscape. The scarcity of annotated data presents a significant challenge for Arabic language models when its compared to English language models as the journey of [13]. The absence of proper annotations hinders the accurate determination of linguistic structures within the text, potentially leading to inaccuracies and biases in the models.

Additionally, due to the absence of capital letters in the Arabic script, distinguishing proper names, titles, acronyms, and abbreviations can be a challenging task [14].

The Arabic language comprises 28 letters and employs eight diacritics, which give rise to a range of phonetic variations in its letters. The accurate placement of diacritics plays a crucial role in discerning word and sentence meanings, leading to the emergence of both morphological and syntactic ambiguities [15].

Moreover, Several factors contribute to the dearth of annotated data for Arabic language models. The intricate nature of the Arabic language, characterized by rich morphology and complex syntax, makes manual annotation a labor-intensive and time-consuming endeavor. Furthermore, the relatively smaller community of Arabic language model researchers, in contrast to languages like English, results in limited resources and funding for extensive annotated corpora creation. These circumstances collectively hamper the growth of Arabic language model research and hinder the development of high-performing models tailored to the language's unique intricacies.

3. Methodology

We took two approaches to text summarization, namely extractive summarization and abstractive. These two approaches have different strengths and weaknesses, which we will discuss below.

Extractive summarization is a more straightforward approach to text summarization. The summarization model will identify the most important sentences in the input text and then create a summary by stitching together these sentences.

Abstractive summarization is a more creative approach to text summarization. The summarization model will read the input text and then generate a new text that is a concise and informative summary of the original text. This approach is often more accurate than extractive summarization, as it can capture the overall meaning of the text in a more comprehensive way. Utilizing pre-trained models we have fine-tuned AraT5 to build an abstractive summarizer

3.1. Description of the Fine-Tuning Process

Fine-tuning is a process of training an existing model that has been trained for another task on a new, specific task or domain. Instead of starting the training model from scratch, fine-tuning leverages the knowledge and learned representations of the pre-existing model to adapt it to the new task. Since NLP models are widely used and publicly available, we have discovered various models that have been trained on a large Arabic corpus, which has significantly advanced natural language understanding and processing for the Arabic language. We utilized AraBERT [16] as a preprocessor for our model (SAraT5).

3.2. Selection of The Pre-Trained Model for Fine-Tuning

We have chosen the AraT5 pre-trained model for fine-tuning for the Arabic text summarization task for several compelling reasons

1. Limited Data Scenario: Fine-tuning a pre-trained model is especially advantageouswhen dealing with limited labeled data for our specific task. Since AraT5 already possesses a broad understanding of the Arabic language, we need less annotated data to fine-tune it for text summarization, making the process more feasible and effective.
2. Performance Boost: Pre-trained models like AraT5 have demonstrated strong performance on various NLP tasks, achieving state-of-the-art results in many cases. By fine-tuning it for text summarization, we expect to benefit from the high-quality features and representations extracted from the pre-training stage, leading to improved summarization performance.

In conclusion, selecting the AraT5 pre-trained model for fine-tuning in our Arabic text summarization task allows us to leverage its Arabic language-specific pre-training, prior experience in text generation tasks, and transfer learning benefits. This choice is well-suited for the limited data scenario.

3.3. Data Collection and Preprocessing

For the first time, we found AMN dataset used in [17] that contains approximately 265k Arabic news with its corresponding summaries. During our exploration, we came across a CNN Mail dataset [18]. comprising human summaries generated from CNN stories. Recognizing the potential of this dataset, we decided to translate it from English to Arabic using the Google Translate API provided in the Python SDK.

After obtaining the translated dataset, we proceeded with data wrangling to address some flaws that emerged during the initial inspection. We identified issues such as duplicated headers throughout both two datasets and instances of redundant occurrences of the word or sentences "CNN," and " èYjJÖÏ@ éJ K. QªË@ H@PAÓB @ (CNN) " among others. These discrepancies were resolved through careful data cleaning and deduplication procedures.

For data preprocessing, we opted to use the Arabert preprocessor, a widely used tool in the NLP community, which is specifically designed for Arabic text. The Arabert preprocessor helped us with essential tasks such as normalization, and handling Arabic-specific linguistic features like diacritics and ligatures. By utilizing this preprocessor, we were able to transform the raw Arabic text into a better format for the model to understand.

3.4. Evaluation Metrics and Techniques Used

The evaluation of the effectiveness and quality of the text summarization models in this research was conducted using Rouge for assessing the performance of summarization approaches. Rouge, which stands for "Recall-Oriented Understudy for Gisting Evaluation," offers a robust framework for measuring the degree of overlap and content alignment between machine-generated summaries and human-written reference summaries. It considers precision, recall, and F1 scores to provide a comprehensive evaluation of how well the generated summaries capture the key content from the source text.

4. EXTRACTIVE APPROACH

Extractive summarization aims to identify and extract the most important sentences or passages from a document, preserving the original context and reducing the length while retaining the essential information. However, before this extraction process can take place, the textual data must undergo several preprocessing steps to ensure accuracy, coherency, and efficiency during vectorization.

4.1. Preprocessing

In the context of extractive text summarization, preprocessing of textual data holds utmost importance as it forms the foundation for transforming raw text into meaningful numerical representations. The Arabic language is complex, with rich and intricate morphological and syntactic flexibility. This makes developing NLP systems for Arabic a challenging task. The preprocessing stage is similar for all languages and typically involves normalization, tokenization, POS tagging, stemming/lemmatization, and stop-word removal. However, the specific implementation of these steps can be more complex for Arabic, due to the language's unique features. The main steps of this approach are Text Cleansing, Stemming, Normalization, Tokenization and Feature extraction.

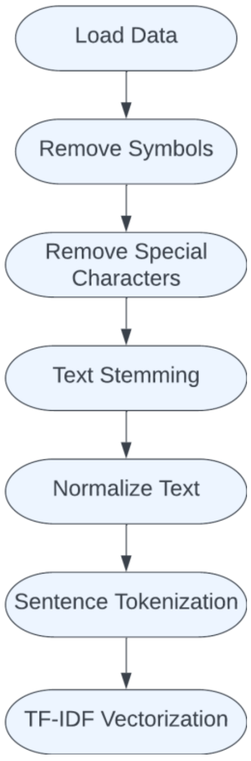

Fig. 1 The pipeline of the preprocessing phase for the extractive approach

4.1.1. Stemming

Arabic is a highly inflectional and derivational language, which means that words can take on many different forms while still retaining their core meaning. This can make it difficult to process

Arabic text using natural language processing techniques, such as text similarity analysis and bag-of-words models.

One way to address this challenge is to use stemming, which is a process of reducing words to their root form. This is done by removing affixes, such as prefixes, infixes, postfixes, and suffixes. Root stemming is a particularly effective technique for Arabic, as it can help to identify words that have the same meaning even though they appear in different forms.

In our study, we used ISIR Arabic stemmer [19] to handle the stemming operation as a preprocessing task. We found that this approach improved the performance of our extractive text summarization system.

4.1.2. Normalization

Arabic is a highly diverse language, with many different ways to write the same word. This can be a challenge for natural language processing (NLP) systems, which need to be able to understand and process text consistently.

Onewaytoaddressthischallengeistousenormalization,whichistheprocessofconverting different forms of the same word into a single, standardized form. This can be done by removing diacritics, replacing similar-looking letters, and eliminating non-Arabic characters.

The PyArabic [20] tool is a popular NLP tool for Arabic. It includes a normalization step that performs the following tasks:

Removes non-Arabic characters, such as special symbols and punctuation. Removes diacritics, which are the small marks that are used to indicate the pronunciation of Arabic letters.
Replace the letters @ , ø ,ð , @ and with Z. These letters are often used interchangeably, but they have different pronunciations. Removes tattoos, which are stretched characters that are sometimes used in Arabic text. Normalization is an important step in the NLP pipeline for Arabic [21]. It helps to ensure that NLP systems can consistently understand and process Arabic text.

4.1.3. Tokenization

Tokenization is the process of dividing text into smaller units, such as words, phrases, or sentences. This is the first step in text preprocessing, and it helps to make text easier to analyze and process.

There are many different ways to tokenize text, and the best approach depends on the specific application. In the context of Arabic text, one common approach is to use the NLTK sentence tokenizer. This tokenizer uses a combination of punctuation marks and whitespace to identify sentences in a document.

4.1.4. Feature Extraction

For the process of feature extraction, we employed the *Term Frequency-Inverse Document Frequency (TF-IDF)* method. This statistical technique holds significance in information retrieval and natural language processing by assessing the significance of a word across a set of documents. In the realm of automatic summarization systems, TF-IDF plays a crucial role in pinpointing significant sentences or phrases within a document, contributing to the creation of concise and purposeful summaries.

4.2. Model Building

1. *Latent Semantic Analysis (LSA)* is a technique used in Natural Language Processing (NLP) and information retrieval to analyze the relationships between words in a large corpus of text. LSA is based on the idea that words with similar meanings tend to occur in similar contexts, and by analyzing these patterns, it can identify latent (hidden) semantic structures and relationships within the text. By that, we utilize latent and TFIDF vectorizer to cluster documents
2. *K-means* is a clustering algorithm that can be used in extractive summarization to group similar sentences or passages together based on their content and similarity. Extractive summarization aims to select a subset of sentences from the original document that can effectively represent the main ideas or important information present in the text. K-means can help achieve this by clustering sentences with similar content by utilizing TFIDF as a vectorizer and then selecting representative sentences from each cluster to form the summary.
3. *K-means with sentence transformer* By incorporating Sentence BERT [22] for sentence vectorization, we observed a significant improvement in the efficacy of our K-means clustering approach. This integration allowed us to represent sentences as highly informative vectors, capturing their semantic nuances more accurately. Consequently, our clustering results became more refined and contextually precise, enhancing the overall quality of our summarization.
4. *Summarizing by using Genetic Algorithms (GA)* This approach tries to utilize genetic algorithms [23] to get the best sentences that best represent the document as a summary.

4.3. Model Evaluation & Comparison

The model evaluation was conducted utilizing the EASC dataset [7]. Comprising 153 Arabic articles, this dataset incorporates 765 human-generated extractive summaries of these articles. These summaries were produced through the employment of Mechanical Turk. The evaluation outcome unmistakably highlights the supremacy of the K-means approaches over LSA across all assessment metrics.

When comparing our three models with the Genetic Algorithms approach, a distinct advantage becomes evident in favor of the Genetic Algorithms approach over the TF-IDF vectorization approaches. However, an interesting observation emerges when we employ the sentence transformer: we achieve similar or slightly higher results. That's an evidence that utilizing old-fashioned TFIDF vectorizer for summarization might not be sufficient with the advancement of transformers.

Table 1 Evaluating the extractive summarization models using ROUGE-1 and ROUGE-L metrics.

Model	ROUGE-1			ROUGE-L		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
LSA - TFIDF	37.04%	38.22%	32.84%	36.41%	37.59%	32.29%
K-Means - TFIDF	**38.82**%	43.48%	36.54%	**38.00**%	42.48%	35.74%
K-Means - Sentence-BERT	37.48%	**50.58**%	**39.17%**	36.75%	**49.70%**	**38.44%**
GA	38.00%	50.00%	39.00%	-	-	-

5. ABSTRACTIVE APPROACH

5.1. Preprocessing

For the Abstractive approach, we performed a series of essential preprocessing steps, including tokenization, and removing diacritics. Arabic text normalization was applied to handle various forms of words and improve consistency. Additionally, we leveraged word embeddings trained on a large corpus to represent Arabic words in a dense vector space, enhancing the model's understanding of the language.

5.2. Implementation

The abstractive summarization model what we Call *SAraT5* was implemented in transformer architecture, Which comprises two main components: the encoder and the decoder. The encoder, based on the AraT5 model, processes the input Arabic text, capturing essential information. The decoder, integrated within the transformer architecture, generates the summary one token at a time, considering the encoded representation of the input text.

Our fine-tuning process involved several key components. To begin, we utilized Python and the "transformers" library by Hugging Face to fine-tune the AraT5 model. This library streamlined our efforts and facilitated the integration of the transformer architecture into our workflow.

5.3. Details of the Fine-Tuning

In this section, we elaborate on the specific details of the fine-tuning process for AraT5, targeting abstractive Arabic text summarization using the CNN and AMN datasets. The optimization process was conducted with careful consideration of various parameters to achieve optimal performance.

5.3.1. Training Duration and Epochs

The fine-tuning process involved training the AraT5 model over 88 hours. This rigorous training procedure encompassed 22 epochs, indicating the number of times the model iterated through the entire training dataset. This extended training duration was employed to ensure thorough convergence and the extraction of nuanced summarization capabilities from the model.

5.3.2. Model Parameters

Several crucial model parameters were meticulously selected to fine-tune AraT5 effectively. These parameters played a pivotal role in shaping the model's behavior and enhancing its summarization prowess for the specific Arabic text summarization task.

Gradient Accumulation Steps: To stabilize the training process and allow the model to accumulate gradients over multiple mini-batches, a gradient accumulation step of 8 was employed. This technique aids in effective parameter updates while managing memory constraints.

Batch Size: A batch size of 8 was chosen for the fine-tuning process. This moderate batch size strikes a balance between efficient parallelization and memory consumption, thereby facilitating steady training progress.

Weight Decay: A weight decay value of 0.01 was applied during training. This regularization technique helps prevent overfitting by introducing a penalty term to the loss function based on the magnitude of model weights.

Learning Rate: The learning rate, a crucial hyperparameter governing the step size in gradient descent optimization, was set at 5e-4. This value was determined through experimentation to ensure steady convergence without causing overshooting or slow convergence issues.

5.4. Model Evaluation & Comparison

In our study, we employed the AMN dataset for evaluation and comparison. Before commencing the training phase, we meticulously partitioned this dataset into two distinct subsets: a training set consisting of 180,000 samples and an evaluation set comprising 5,000 samples. The latter, namely the 5,000-sample evaluation dataset, served as a critical component in our study's evaluation framework. To facilitate meaningful comparisons with prior research in the domain of Arabic text summarization, we did a comparison with the work of [8] which is a sequence-to-sequence model with BiLSTM layer, which focused on Abstractive Arabic Text Summarization Based on Deep Learning.

Table 2 Comparing the abstractive summarization model (SAraT5) using ROUGE-1, ROUGE-2, and ROUGE-L

Model	ROUGE-1			ROUGE-2			ROUGE-L		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
SAraT5 (Proposed Model)	35.11%	**58.98%**	42.29%	**24.90%**	**43.68%**	**30.18%**	33.60%	**56.17%**	**40.40%**
Seq2Seq - BiLSTM	**48.15%**	42.65%	**44.28%**	19.46%	17.93%	18.35%	**35.48%**	32.86%	32.46%

6. DISCUSSION

6.1. Interpretation of the Results and their Implications

The results obtained from our experimental analysis provide valuable insights into the performance of both the extractive and abstractive summarization approaches in the context of Arabic text. In the extractive approach, we observed that K-means clustering outperformed Latent Semantic Analysis (LSA) in terms of the ROUGE metrics, indicating its effectiveness in identifying coherent and meaningful sentences for summary generation. This result suggests that tailored methods, such as K-means clustering, can yield more accurate and representative extractive summaries, allowing us to capture the essence of the source text more effectively.

6.2. Limitations of the Proposed Approach

For the Extractive Summarization approach, it introduces a limitation particularly relevant to relatively large documents. The inherent constraint of generating a highly concise summary, composed of only three sentences, can result in a potential loss of contextual information and depth. This limitation is especially notable in the context of larger documents where a broader scope of content may be necessary to adequately capture the nuances and comprehensive essence of the source material.

On the other hand, the abstractive approach *SAraT5* model presented some challenges. Despite its strong performance in summarizing articles, it faced challenges in summarizing other document categories such as medical prescriptions.

6.3. Possible Extensions and Future Research Directions

Future research could focus on improving the coherence, fluency, and factual accuracy of abstractive summaries. Developing techniques that better understand the nuances of the Arabic language, including its intricate grammar and semantics, would be beneficial.

Also, Designing models that can specialize in summarizing specific domains, such as medical literature, legal documents, or news articles, could lead to more accurate and informative summaries. Fine-tuning models on domain-specific data and creating specialized datasets for Arabic could help in this direction.

7. CONCLUSION

7.1. Summary of the Research Findings

The evolution of Large Language Models (LLMs) has catalyzed the integration of NLP-related technologies, bringing us closer to machines that can comprehend and generate human-like text.

The extractive and abstractive summarization approaches shed light on how NLP can facilitate the generation of concise and informative summaries. Through the evaluation using the EASC dataset, the dominance of the K-means approach over Latent Semantic Analysis (LSA) in extractive summarization becomes evident, reinforcing the value of tailored techniques in language-specific tasks.

The abstractive approach has the performance of state-of-the-art models in the realm of Arabic summarization on AMN dataset.

7.2. Contribution to the Field of Arabic Language Processing and NLP

This research significantly contributes to the field of Arabic language processing and natural language processing (NLP) by addressing the specific challenges and opportunities that arise when working with Arabic text. The exploration of both extractive and abstractive summarization techniques offers a comprehensive perspective on summarization methods that are well-suited for Arabic content. The fine-tuning of the AraT5 model for abstractive summarization in Arabic, although challenging, highlights the ongoing need for language-specific adaptation in NLP.

Moreover, this research underscores the importance of dataset availability and quality in advancing Arabic NLP. By presenting our methodologies and findings, we hope to encourage collaboration between researchers, academia, and industry stakeholders to further enrich Arabic NLP resources. The advancements made in this study pave the way for more accurate, culturally sensitive, and contextually aware summarization systems that cater to the Arabic language's unique linguistic characteristics and cultural nuances.

REFERENCES

[1] C.-Y. Lin, “ROUGE: A package for automatic evaluation of summaries,” in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https: //aclanthology.org/W04-1013

[2] E.M.B.Nagoudi, A.Elmadany, and M.Abdul-Mageed,“Arat5:Text-to-texttransformers for arabic language generation,” 2022.

[3] M.Al-Maleh and S.Desouki, “Arabic text summarization using deep learning approach,” *Big Data 7, 1–17*, pp. 152628–152645, 2020.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019.

[5] Q. Al-Radaidehand D.Bataineh,“A hybridapproachforarabic textsummarizationusing domain knowledge and genetic algorithms,” *Cognitive Computation*, vol. 10, 08 2018.

[6] M. El-Haj and R. Koulali, “El-haj, m., koulali, r. "kalimat a multipurpose arabic corpus" at the second workshop on arabic corpus linguistics (wacl-2) 2013,” 01 2013.

[7] M. El-Haj, U. Kruschwitz, and C. Fox, “Using mechanical turk to create a corpus of arabic summaries.” France: European Language Resources Association, 2010, published proceedings: _not provided_ - Notes:. [Online]. Available: https://repository.essex.ac.uk/4064/

[8] A. A. A. A. A. Y. M. Wazery, Marwa E. Saleh, “Abstractive arabic text summarization based on deep learning,” pp. 152628–152645, 2022. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8767398/pdf/CIN2022-1566890.pdf

[9] D. Suleiman and A. Awajan, “Multilayer encoder and single-layer decoder for abstractive arabic text summarization,” *Knowledge-Based Systems*, vol. 237, p. 107791, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/ S0950705121010005

[10] A. B. Ayed, I. Biskri, and J.-G. Meunier, “Arabic text summarization via knapsack balancing of effective retention,” *Procedia Computer Science*, vol. 189, pp. 312–319, 2021, aI in Computational Linguistics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050921012242

[11] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, and S. Paul, “Peft: State-of-the-art parameter-efficient fine-tuning methods,” https://github.com/huggingface/peft, 2022.

[12] A. Parnami and M. Lee, “Learning from few examples: A summary of approaches to few-shot learning,” 2022.

[13] R. Obiedat, D. Al-Darras, E. Alzaghoul, and O. Harfoushi, “Arabic aspect-based sentiment analysis: A systematic literature review,” *IEEE Access*, vol. 9, pp. 152628–152645, 2021.

[14] A. Azmi and N. Altmami, “An abstractive arabic text summarizer with user controlled granularity,” *Information Processing and Management*, vol. 54, pp. 903–921, 11 2018.

[15] A. Azmi, “A survey of automatic arabic diacritization techniques,” *Natural Language Engineering*, vol. 21, 10 2013.

[16] W. Antoun, F. Baly, and H. Hajj, “AraBERT: Transformer-based model for Arabic language understanding,” in *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*. Marseille, France: European Language Resource Association, May 2020, pp. 9–15. [Online]. Available: https://aclanthology.org/2020.osact-1.2

[17] A. M. Zaki, M. I. Khalil, and H. M. Abbas, “Deep architectures for abstractive text summarization in multiple languages,” in *2019 14th International Conference on Computer Engineering and Systems (ICCES)*, 2019, pp. 22–27.

[18] C. d. S. G. u. B. X. Ramesh Nallapati, Bowen Zhou, “Arabert: Transformer-based model for arabic language understanding,” 2016.

[19] D. Abd, W. Khan, K. Thamer, and A. Hussain, *Arabic Light Stemmer Based on ISRI Stemmer*, 08 2021, pp. 32–45.

[20] T. Zerrouki, “pyarabic, an arabic language library for python,” 2010. [Online]. Available: https://pypi.python.org/pypi/pyarabic

[21] R. M., H. Mousa, and M. Hussein, “Improving arabic text categorization using normalization and stemming techniques,” *International Journal of Computer Applications*, vol. 135, pp. 38–43, 02 2016.

[22] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Association for Computational Linguistics, 11 2019. [Online]. Available: http://arxiv.org/abs/1908.10084

[23] I. Tanfouri, G. Tlik, and F. Jarray, "An automatic arabic text summarization system based on genetic algorithms," *Procedia Computer Science*, vol. 189, pp. 195–202, 2021, aI in Computational Linguistics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705092101200X