Republic of Algeria Democratic and Popular
Ministry of Higher Education and Scientific Research
University of Abou Bekr Belkaïd - Tlemcen
Faculty of Sciences
Department of Computer Science

FINAL YEAR PROJECT MASTER THESIS

For the attainment of the Master's degree in Computer Science

Specialty: Intelligent and Decision Models (M.I.D)

Theme:

Creation of an automatic summary tool for arabic texts

Presented on September 30th, 2024, before the jury composed of:

- **Dr. Benziane Yaghmoracen** — President
- **Dr. Belabed Amine** — Examiner
- **Dr. Abderrahim Alaeddine** — Supervisor

Realized by:

- **Mr. Taleb Ben Diab Bashir**

Academic Year: 2023 / 2024

Words of Gratitude

First and foremost, I thank God the Almighty for granting me the strength, perseverance, and determination to complete this project. Without His blessings, none of this would have been possible.

I would like to extend my sincere gratitude to my supervisor, Mr. Abderrahim Mohammed Alaeddine. Your invaluable guidance, continuous support, and insightful feedback have been instrumental in shaping the direction and outcome of this work. I truly appreciate your dedication and the time you devoted to helping me succeed.

I would like to extend my heartfelt gratitude to my dear parents, whose unwavering love, encouragement, and sacrifices have been my foundation. Your constant support, both emotionally and practically, has sustained me throughout the most difficult moments. You have always believed in me, even when I doubted myself, and for that, I am eternally grateful. I dedicate this accomplishment to you, as none of this would have been achievable without your love, guidance, and endless patience.

To my family and friends, thank you for your boundless encouragement and understanding during this demanding journey. Your words of reassurance, your presence in moments of need, and the joy you brought into my life have given me the energy to continue when times were tough. You were always there to lift my spirits, whether through kind words, a listening ear, or just by being there when I needed support. I am deeply appreciative of your unwavering faith in me.

I am also deeply thankful to the esteemed jury members who have graciously agreed to evaluate this project. Your expertise, time, and constructive suggestions are genuinely appreciated and have helped elevate the quality of this work.

Finally, I express my heartfelt appreciation to everyone who has contributed to this project, whether directly or indirectly. Each of your contributions, no matter how large or small, has made a significant impact on my academic journey and the completion of this project.

Content Table

List of Figures

General Introduction

1. Context

In recent years, natural language processing (NLP) has seen remarkable advancements, particularly in the domain of text summarization. As the volume of information in digital form continues to grow, tools for automatic summarization have become essential for extracting relevant content from large text corpora. Summarization models play a pivotal role in helping users sift through data, delivering concise versions of the original text while preserving its essential information. This is particularly relevant for languages such as Arabic, which, due to its unique linguistic structure, presents challenges for NLP. Although considerable progress has been made in languages like English, the development of robust models for Arabic is still in its nascent stage. This thesis aims to address these gaps by contributing to the Arabic text summarization landscape.

2. Problem Statement

Despite the growing need for automatic summarization, existing solutions for Arabic texts remain underdeveloped compared to those for English and other widely spoken languages. Available models often face issues such as limited training datasets, lack of specialized pre-trained models for Arabic, and challenges in tokenization due to the complexities of the language. These limitations result in suboptimal performance in terms of accuracy, fluency, and relevance of generated summaries. Additionally, cross-platform accessibility and user-friendly interfaces for these tools are often neglected. The gap in reliable, efficient, and accessible summarization tools for Arabic texts creates a demand for further research and model development in this area.

3. Our Motivation

Our motivation stems from the need to improve Arabic language processing, particularly in summarization tasks. The scarcity of high-quality, open-source tools that can cater to various Arabic dialects and Modern Standard Arabic (MSA) underscores the need for innovation. Furthermore, Arabic texts, being more syntactically and morphologically complex than many other languages,

pose unique challenges that existing models have not adequately addressed. We seek to contribute by developing a web-based solution that not only generates high-quality summaries but also offers a practical, user-friendly platform for Arabic speakers and researchers alike.

4. Contribution

This thesis offers multiple contributions to the field of Arabic NLP, particularly in text summarization:

- **Comparative Study**: A comparative analysis of existing state-of-the-art models on Arabic texts, focusing on performance metrics such as ROUGE and BERTScore.
- **Fine-tuned Model**: The presentation of a fine-tuned model specifically trained on Arabic datasets, with the aim to surpass the performance of existing models.
- **Web Application**: Deployment of the solution as a web-based tool accessible across platforms, focusing on providing Arabic text summarization. This improves accessibility and offers a practical tool for Arabic-speaking users and NLP researchers.

5. Manuscript Outline

The thesis is organized into four main parts, starting with a general introduction, followed by two detailed chapters on related work and our contribution, and concluding with a general conclusion.

- **General Introduction**: This section sets the stage for the thesis, providing the background and context of the research. It explains the problem statement, our motivation, and the specific contributions we aim to make through this work. It also provides a brief overview of the manuscript structure.
- **Chapter 1: Related Work**: This chapter delves into existing models, techniques, and research related to Arabic text summarization. It includes a critical analysis of methods, highlighting their strengths and limitations, along with topics such as tokenization approaches, evaluation metrics, and challenges in Arabic NLP.

- **Chapter 2: Contribution**: This chapter focuses on our original contributions, covering the tokenization process, fine-tuning of models, and data preprocessing steps. It also presents the architecture and features of the developed web application, discussing both its strengths and potential areas for improvement.
- **General Conclusion**: The thesis concludes with a summary of findings, discussing their significance and overall contributions to Arabic NLP. Future research directions and potential optimizations of the summarization tool are also addressed.

Chapter 1: Related Work

1. Introduction

Text summarization has significantly evolved due to advancements in neural networks and Natural Language Processing (NLP). Traditional methods like extractive summarization have been augmented by abstractive techniques, thanks to architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformer-based models like BERT and GPT. While summarization for English has shown substantial progress, languages like Arabic introduce specific challenges related to morphology, syntax, and data scarcity. This chapter explores existing models, the role of NLP, and highlights the unique challenges of summarizing Arabic texts, supported by research.

2. Traditional Text Summarization Approaches

- **Extractive Summarization:**
  Extractive methods, which select key phrases or sentences, have been widely used in many languages, including Arabic. However, the morphological richness of Arabic complicates such methods ,[1] extracting key sentences from Arabic texts can fail to capture the language's complex syntactic structures, leading to a lack of nuance in summaries. Recent research highlights the challenges of applying extractive methods in morphologically rich languages like Arabic [2].
- **Abstractive Summarization:**
  Abstractive summarization generates new sentences based on the source material, offering a more flexible approach but demanding deeper linguistic understanding. Arabic, with its flexible syntax and complex morphology, complicates this process. There has been research [2] that notes that neural approaches have improved abstractive summarization in Arabic, but issues like word-order flexibility and ambiguity remain.

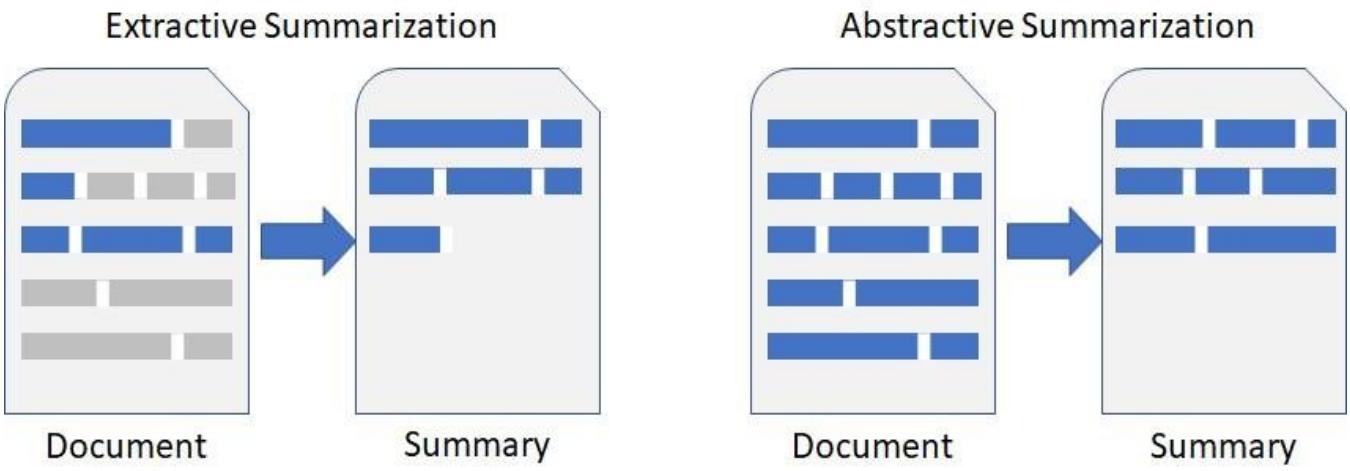

Fig 1: An image contrasting extractive vs. abstractive summarization models.

3. Neural Network Models for Summarization

- **Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM):**
  RNNs and LSTMs have played a critical role in sequence-based tasks, including Arabic text summarization. Their ability to capture sentence-level dependencies makes them more effective than traditional extractive methods. However, as pointed out in this article [2], RNN-based models struggle with longer Arabic texts due to the language's morphological complexity, which leads to tokenization challenges.

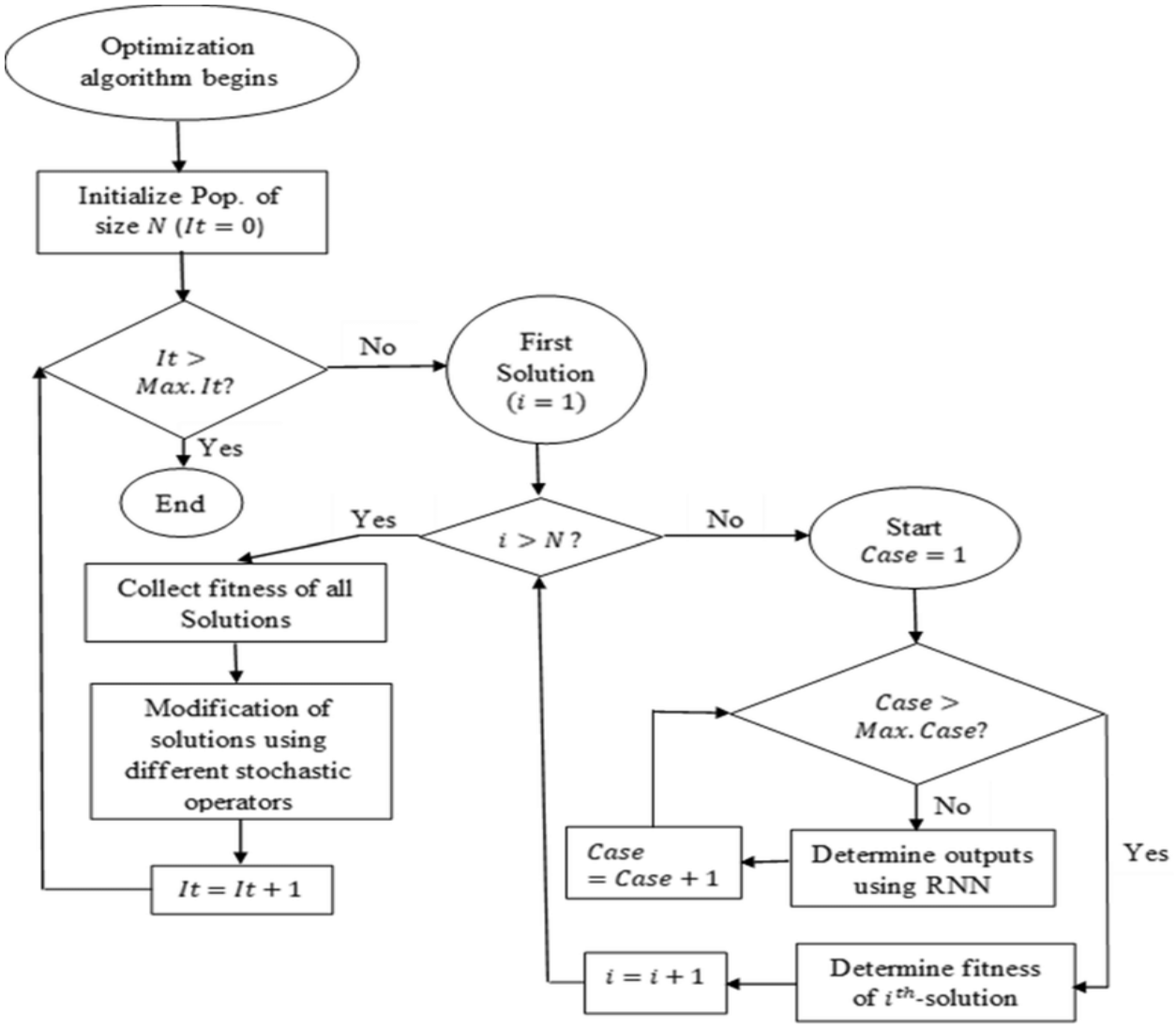

Fig 2: Flowchart of recurrent neural networks tuned by an optimization algorithm.

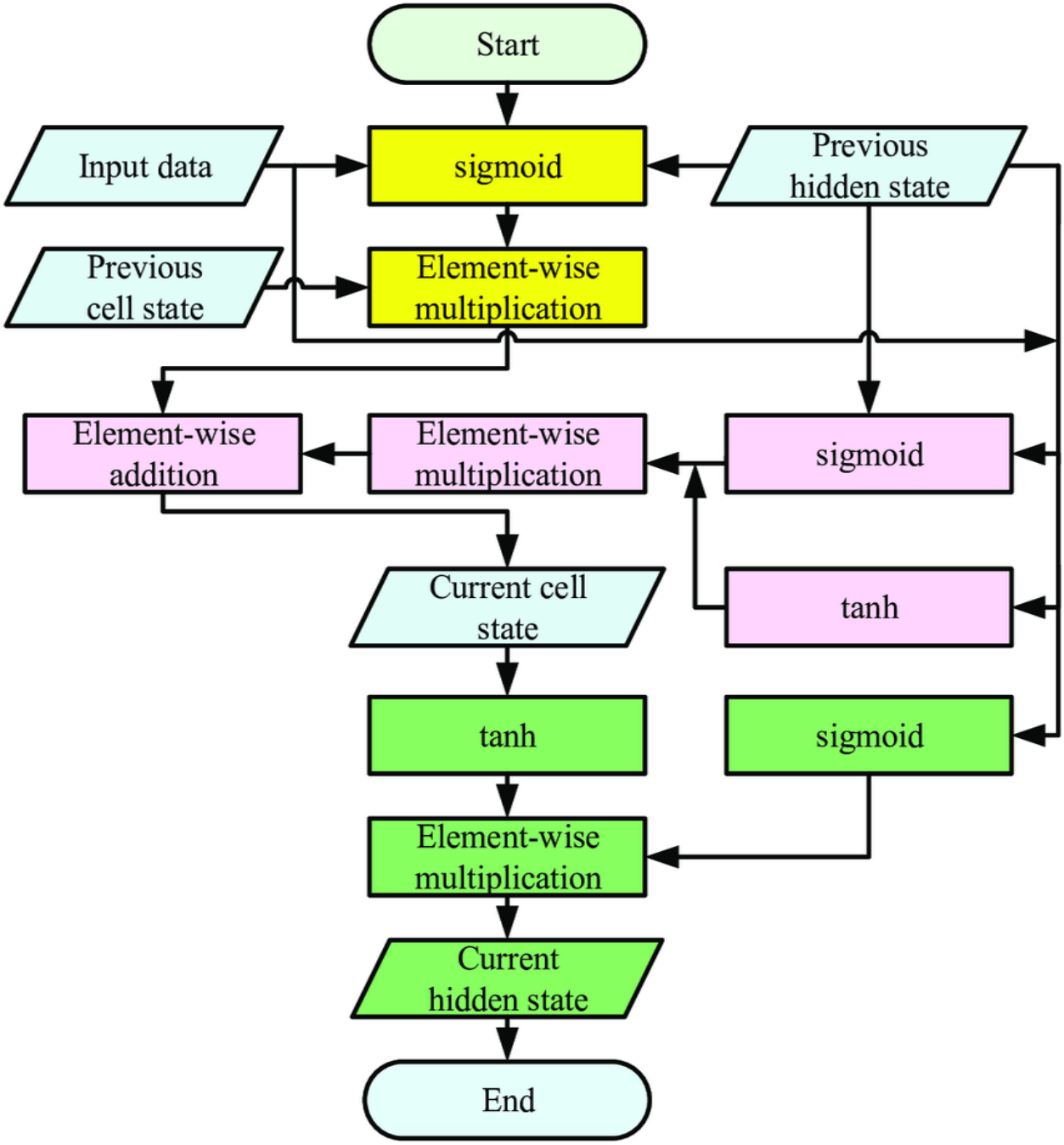

Fig 3: Flow chart of LSTM's operation. The yellow, pink and green blocks indicate the forget, input and output gates, respectively.

- **Convolutional Neural Networks (CNNs):**
  While CNNs have traditionally been applied to image recognition tasks, their ability to capture local features has been adapted for summarization. CNN-based approaches have shown promise in Arabic extractive summarization, though their application remains limited for abstractive tasks [3]. These models offer efficient solutions for shorter texts but tend to perform poorly when tasked with summarizing long Arabic documents.

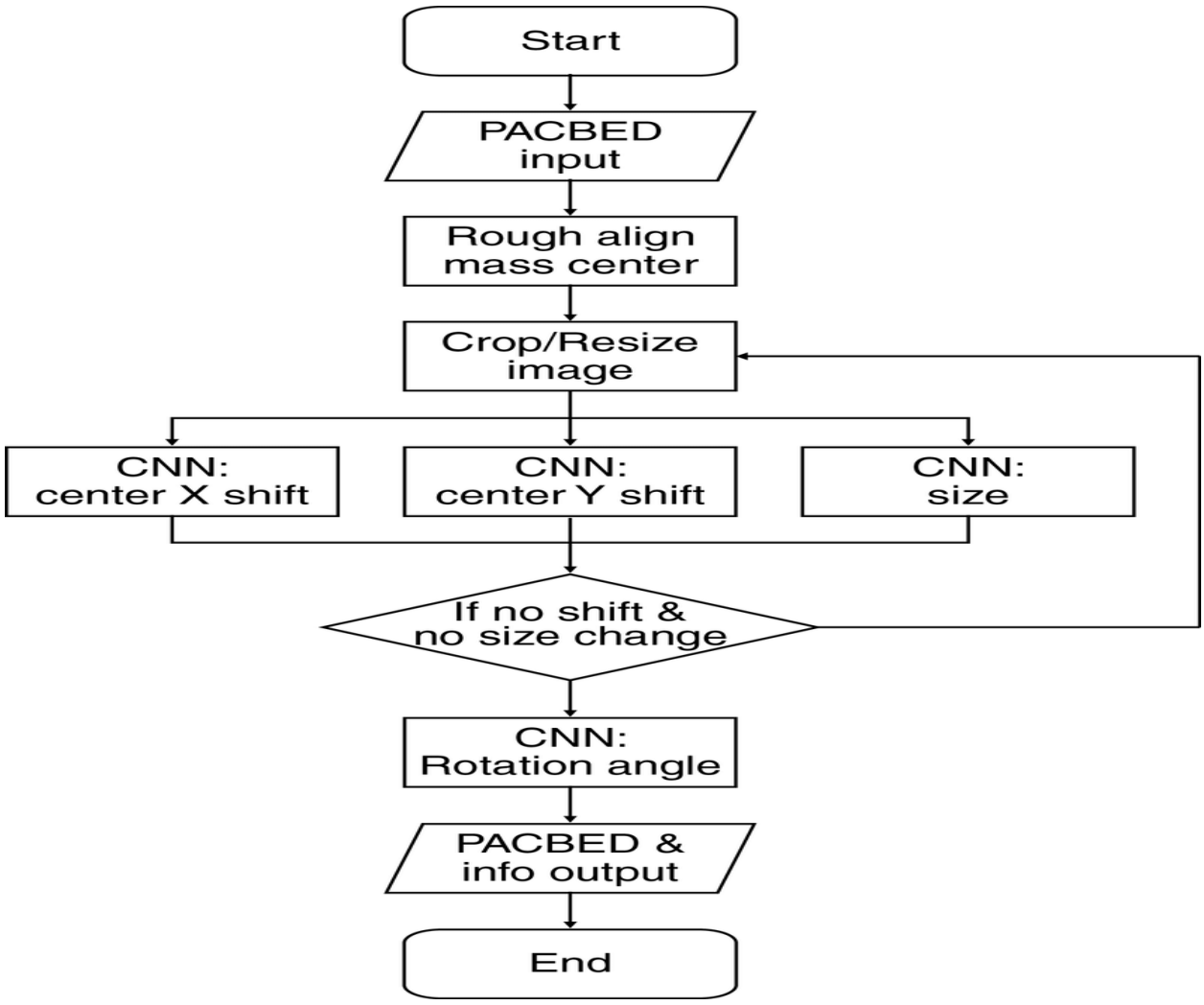

Fig 4: Flow chart of the convolutional neural networks implemented during an automated alignment procedure.

- **Transformer-based Models:**
  Transformers have revolutionized text summarization by using self-attention mechanisms to process long-distance dependencies in a text. Models like T5, BERT, and GPT have shown promising results for Arabic summarization. In their work, [4] this research highlights that fine-tuning BERT specifically for Arabic has significantly improved the quality and coherence of generated summaries, though challenges remain due to a lack of high-quality Arabic datasets.

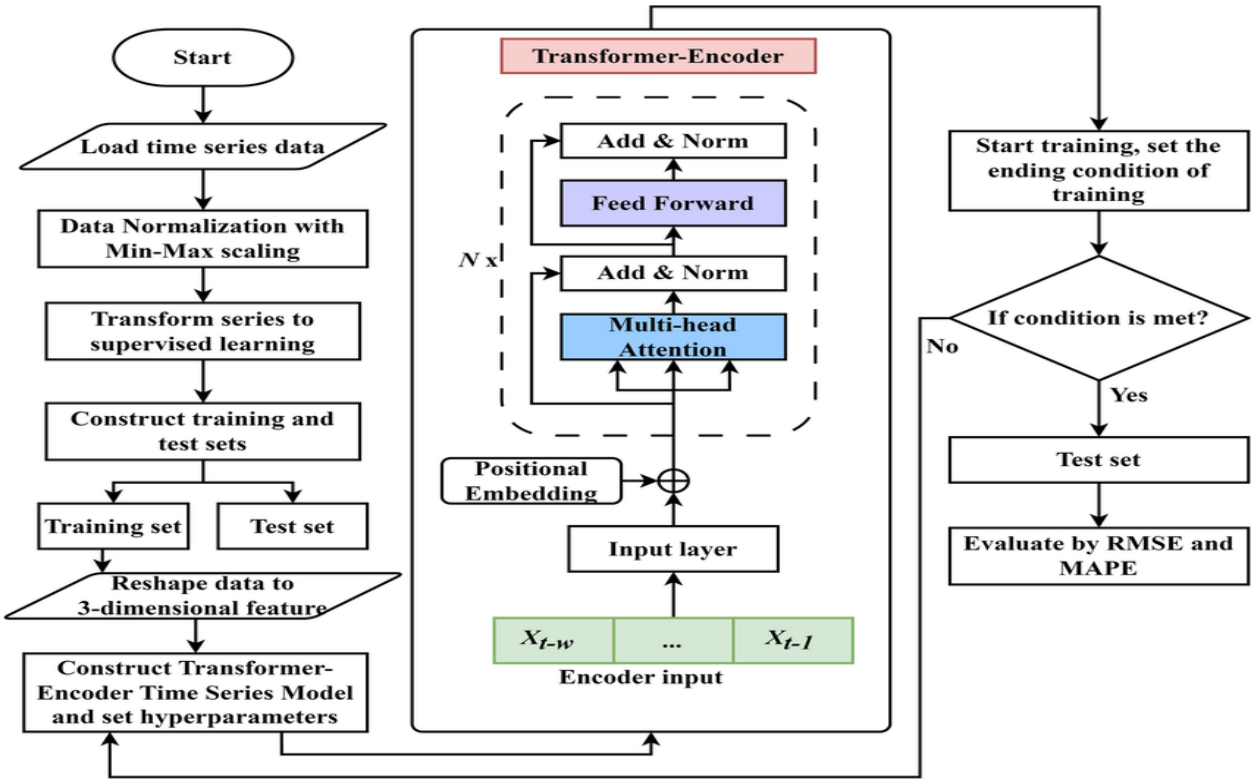

Fig 5: A flowchart of a Transformer-based model.

4. NLP and Its Use in Summarization

Natural Language Processing (NLP) is fundamental to both extractive and abstractive summarization tasks, as it enables machines to understand, process, and generate human language. In Arabic text summarization, NLP techniques like tokenization, part-of-speech tagging (POS), and named entity recognition (NER) are crucial due to the complexity of the Arabic language.

- **Tokenization and POS Tagging:**
  Tokenizing Arabic is challenging because of its rich morphology and root-based structure. According to this research [4], the tokenization process is critical in handling Arabic's complex affixes, which, if not properly processed, lead to inaccuracies in summarization. POS tagging also plays an essential role in identifying the grammatical structure of sentences, improving both extractive and abstractive summarization [3].
- **Named Entity Recognition (NER):**
  NER is vital for summarization, ensuring the meaning of key entities is preserved. In Arabic, NER faces additional challenges due to linguistic variability, but techniques explored by [3] Al-Ghamdi et al. (2020) show that NER models significantly enhance the quality of summaries, particularly in capturing proper nouns and other critical entities.
- **NLP in Transformer Models:**
  NLP techniques empower Transformer-based models like BERT and T5 to handle long-range dependencies and contextual meanings in text. These models have substantially improved Arabic summarization, as demonstrated by [1] this research, where fine-tuning BERT for Arabic significantly improved the coherence of generated summaries.

5. Challenges in Summarizing Arabic Text

Arabic introduces unique challenges for summarization due to:

- **Rich Morphology:** Arabic's root-based system, which allows for the generation of multiple word forms, complicates tokenization and processing. [2] This research notes that this morphological richness presents difficulties for neural networks in understanding and generating coherent summaries.
- **Flexible Word Order:** Arabic syntax allows for greater flexibility in sentence structure than English. This flexibility can confuse models trained on more rigid languages, as indicated in the work of [3] Habash (2010).
- **Scarcity of Annotated Data:** The lack of large annotated datasets in Arabic is a major obstacle for developing efficient summarization models. Researchers [9] have shown that applying transfer learning from English models to Arabic tasks can mitigate some of these issues, though it is not a perfect solution.

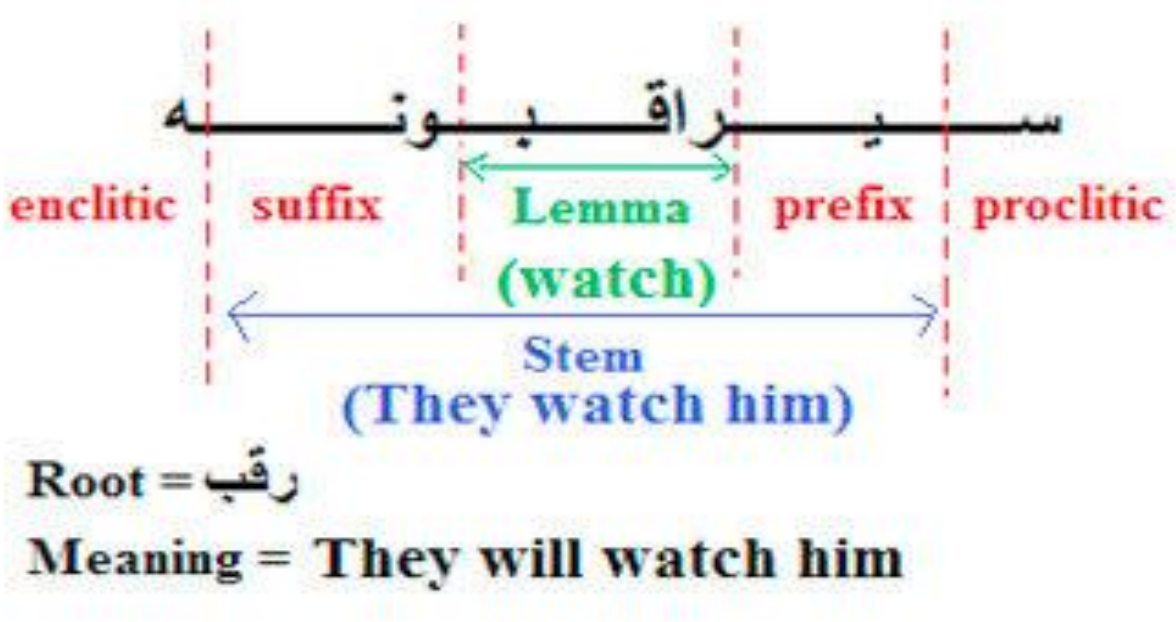

Fig 6: An image illustrating Arabic morphological complexity.

6. Challenges in Current Models

- **Computational Efficiency:**
  Neural models, especially those based on Transformers, are known for being computationally demanding, and this challenge becomes even more pronounced when dealing with Arabic text due to the additional preprocessing required to handle its morphological and syntactic complexity. Arabic's rich structure demands more computational power for tokenization and syntactic parsing, leading to increased resource consumption. According to this research [1], the cost of computation rises significantly when scaling Transformer-based models to real-time applications for Arabic text summarization.
- **Factual Consistency and Coherence:**
  While abstractive models tend to generate more fluent summaries, they sometimes suffer from factual inaccuracies, particularly when summarizing specialized domains such as legal or medical texts. This issue is exacerbated in Arabic due to the linguistic flexibility of the language.

This research [2] discusses how Transformer-based models, despite their advances, can generate summaries that are semantically inconsistent, especially when handling complex, domain-specific Arabic texts.

- **Scalability:**
  Large models such as BERT and GPT face significant challenges when scaling to real-time or large-scale Arabic summarization tasks. Handling large datasets or processing high volumes of text in real-time often leads to performance bottlenecks. The work of Al-Ghamdi et al. (2020) [3] highlights how these models struggle with scalability, suggesting that developing more lightweight, efficient models tailored specifically for Arabic could provide a solution to this issue. These specialized models could improve both the scalability and efficiency of summarizing Arabic texts

7. Conclusion

This chapter reviewed the evolution of text summarization techniques, focusing on the challenges specific to the Arabic language. From traditional extractive methods to neural network-based approaches, the complexities of Arabic morphology and syntax continue to challenge summarization models. While Transformer-based models have advanced the field, issues of scalability, computational efficiency, and factual consistency persist. These challenges form the basis for the contributions proposed in the next chapter, which aim to improve the efficiency and accuracy of Arabic text summarization.

Chapter 2: Our Contribution

1. Introduction

In this chapter, we detail the steps taken to develop an Arabic text summarization system. Our contribution includes the design and fine-tuning of a T5 model, the pre-processing of Arabic text, and the construction of a user-friendly web application to interact with the summarizer. We also compare our model's performance with existing Arabic summarization systems using standard evaluation metrics like ROUGE and BERTScore.

2. Tools used

- **Lightning AI Platform:** This platform offers a seamless environment for deep learning workflows and provides various advantages and features like portability and ease of use between multiple machines,but most important of these features in free cloud GPU like the Nividia T4 an L4, the platform offers new users 15 tokens when the account is first created and offers 15 more after every 30 days.
  In this project, the Nvidia L4 GPU was used through this platform because of its massive 24 GB of VRAM which helped reduce training time significantly and its low tokens consumption of 0.52 Tokens/h at the time of making this app.

- **VS Code:** Used for being a lightweight and fast IDE that offers extensive customization through a vast library of extensions, making it versatile for various programming languages. Its integrated terminal enhances workflow efficiency by allowing command-line access without leaving the editor, while powerful debugging tools simplify the process of identifying and fixing issues. VS Code also features seamless integration with Git and other version control systems, facilitating collaboration and code management. With an intuitive interface, it is accessible for beginners and

provides advanced features for experienced developers, making it a popular choice among developers for diverse projects.

- **Hugging Face :** an excellent platform for learning AI skills. It offers a comprehensive set of tools and resources for training and using models. This includes demos, use cases, documentation, and tutorials that guide you through the entire process of using these tools and training models. It was also used for the selection of the T5-base model and the AGS-Corpus dataset used in this project.

3. Technologies used

The following programming languages and technologies were used in the development of the Arabic text summarization application:

- **Python**: For building the back-end, implementing the summarization model, and handling the fine-tuning of the T5 model.
- **HTML/CSS**: Used to structure and style the front-end interface within the Streamlit framework.
- **JavaScript**: Included for front-end interactivity in Streamlit's built-in functionality.
- **Streamlit**: A Python-based framework used to develop the web application interface.
- **PyTorch**: Used for model training and handling deep learning operations.

4. Datasets Used

The dataset used for training the model in this project was the **AGS-Corpus**, which was sourced from **Hugging Face's dataset repository**. This corpus is designed specifically for Arabic summarization tasks, making it an ideal choice for fine-tuning the **T5-base model** for this project.

For efficient training and to reduce computational overhead, the **AGS-Corpus** dataset was partially sampled. Using the Hugging Face datasets library, the training set was shuffled with a seed of 42 for reproducibility, and a **50%** sample of the dataset was selected to be used in the training process.

After sampling, the dataset was split into two subsets:

- **80%** of the sampled dataset was allocated for training the model.
- **20%** was reserved for evaluation and testing purposes.

5. Model and Tokenizer Used

In this project, we utilized Google's T5-base model and its corresponding tokenizer. The T5 (Text-To-Text Transfer Transformer) model is a pre-trained sequence-to-sequence transformer model developed by Google that converts all NLP tasks into a text-to-text format. For our summarization task, the T5 model is fine-tuned to handle Arabic text, where the input is an Arabic document or article, and the output is a concise summary of the input text.

6. T5-base Overview:

- **Model Architecture**: The T5-base model consists of 12 layers, each comprising a self-attention mechanism and feed-forward neural networks. It contains 220 million parameters, making it large enough to understand complex linguistic patterns and relationships between tokens in a sequence.
- **Pre-training**: T5 was pre-trained on a diverse corpus using a denoising autoencoder objective, where part of the input text is corrupted, and the model learns to reconstruct the original text. This pre-training enhances the model’s ability to generalize across various tasks, including summarization.

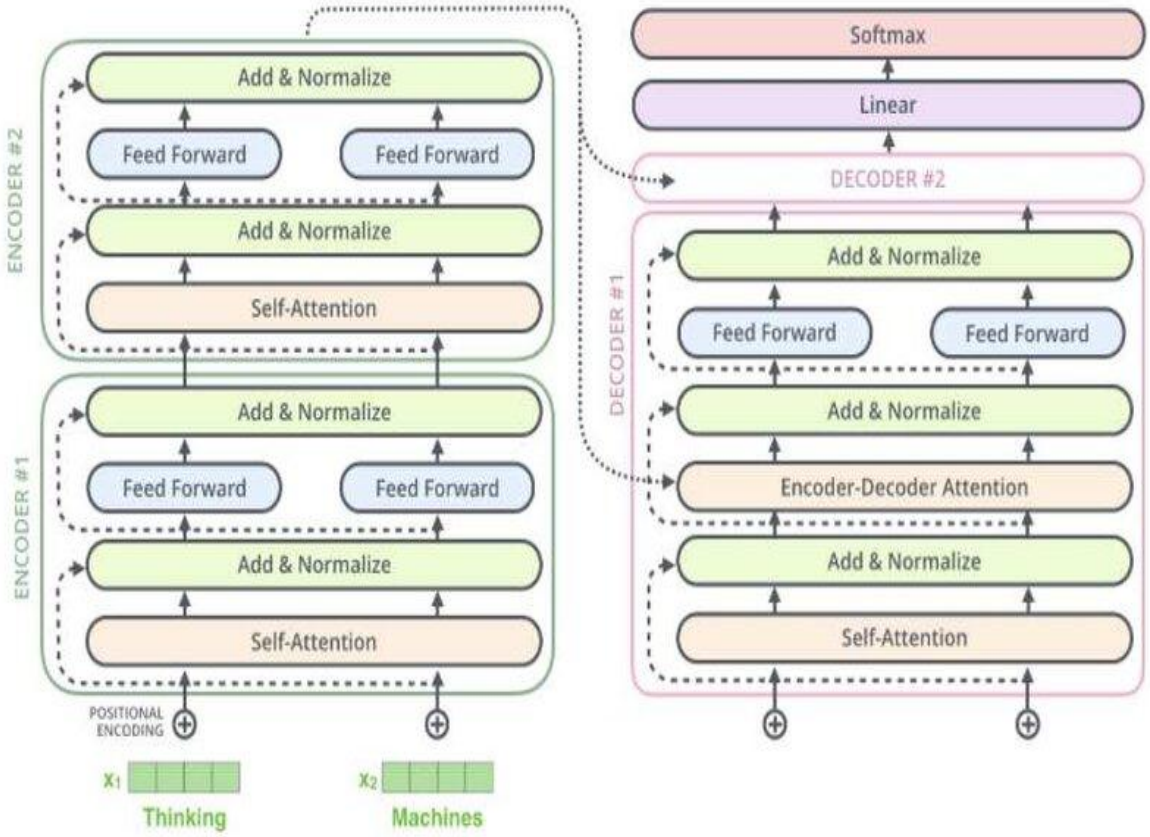

Fig 7: A diagram detailing the architecture of the T5: text-to-text transfer transformer .

7. Tokenizer:

The T5 tokenizer, built on a SentencePiece model, efficiently handles tokenization for multiple languages, including Arabic. It splits text into subwords, allowing for the effective handling of the morphological richness present in Arabic, where words can have many inflected forms.

For this application, the T5-base model was fine-tuned to summarize Arabic text accurately. The tokenizer processes the input text, breaking it down into manageable tokens for the model, ensuring the input remains within the model's predefined token limit, which is essential when handling longer texts like articles. The model then generates a summary based on the tokenized input, which is further processed into human-readable text by the tokenizer.

By leveraging the strengths of the T5 architecture, this project achieves efficient and high-quality summarization of Arabic content, even with its inherent linguistic complexity.

8. Pre-processing the Dataset:

Before feeding the data into the T5 model, pre-processing is essential to ensure that the text is in a format the model can effectively interpret. The pre-processing pipeline for this Arabic text summarization project primarily involves tokenization, handling text length, and ensuring compatibility with the T5-base model.

- **Tokenization and SentencePiece**:
  Google's T5 model uses a SentencePiece tokenizer, which is designed to handle languages with complex morphology, like Arabic. SentencePiece divides the text into subword units, ensuring that even words with prefixes, suffixes, or infixes are split effectively. This helps to handle the morphological richness of Arabic, as indicated by research on subword tokenization methods used in neural networks.
- **Padding and Truncation**:
  Standard practice when training models like T5 involves padding sequences to a uniform length and truncating sequences that exceed a pre-set maximum length. This is critical for efficient computation on GPUs, where tensors must have consistent dimensions. Models like T5 have maximum input length constraints, hence the need for this step.
- **Special Tokens**:
  Special tokens (such as `<pad>`, `<eos>`, `<sep>`) are a built-in feature in most modern transformer-based tokenizers like T5. They signal the start or end of sequences and help distinguish between the input text and target output. This is essential for models like T5, which are sequence-to-sequence models that expect clearly defined input-output structures.
- **Normalization**:
  Text normalization, especially for languages like Arabic, is a well-established pre-processing step. Given the orthographic variations (such as different forms of the same letter, the use of diacritics, and punctuation variations), it is crucial to standardize the text for better model performance. Many papers on Arabic NLP techniques highlight the importance of removing diacritics and normalizing characters.

For the first batch of 10 epochs in this project, the **Google T5-base Tokenizer** was used for both input texts and target summaries. The `AutoTokenizer` from Hugging Face's `transformers` library was utilized, specifically without the `use_fast` option, to allow for a more controlled and thorough tokenization. SentencePiece, which the tokenizer is based on, handled the tokenization process effectively.

Key steps involved in the pre-processing phase:

- The input text (the document to be summarized) and the target summary were extracted from the dataset.
- Each input was tokenized with a maximum length of 300 tokens to avoid truncation and to fit within the model's input limits.
- Similarly, target summaries were tokenized with a maximum length of 100 tokens, as the summarization model typically outputs shorter texts.
- During tokenization, padding was used to ensure that all sequences have the same length, a requirement for batch processing.

For the second batch of 10 epochs,the same pattern was followed as the previous 10 epochs but the slight changes in maximum input and target length as they were increased to 400 and 125 respectively.

These pre-processing steps ensure that the dataset is optimized for fine-tuning the T5 model, allowing for more accurate and efficient training and summarization. The pre-processed text can then be fed into the model, allowing it to effectively generate summaries of Arabic text with a high degree of linguistic complexity.

9. Fine-Tuning Process

The fine-tuning of the T5 model for Arabic text summarization was carried out in two phases, each consisting of 10 epochs. This structured approach aimed to enhance model performance while addressing potential tensor issues during training.

- **Phase 1: Initial Fine-Tuning**

In the first phase, a custom `Seq2SeqTrainer` was implemented to handle non-contiguous tensors. The training arguments were defined to configure various aspects of the training process, such as the learning rate, batch size, and checkpoint saving strategy.

Key parameters for this phase included:

- **Learning Rate**: 3e-5
- **Batch Size**: 8 (both training and evaluation)
- **Gradient Accumulation Steps**: 2
- **Evaluation Steps**: Every 500 steps
- **Total Number of Epochs**: 10

The model and tokenizer were trained and saved for future use.

- **Phase 2: Advanced Fine-Tuning**

In the second phase, the model was further fine-tuned using the previously saved model and tokenizer. The same custom trainer was utilized to ensure consistency in handling tensor issues.

The training arguments were slightly adjusted for this phase:

- **Learning Rate**: 5e-5 This phase followed a similar structure as the first maintaining the overall training strategy and logging mechanisms.

10. Obstacles encountered and solutions around it

During the development of the Arabic text summarization system, several obstacles were encountered that required both technical adjustments and methodological decisions to be made.

- **ValueError with Non-Contiguous Tensors**
  A significant challenge arose during the model's fine-tuning phase. Specifically, a `ValueError` occurred when attempting to save non-contiguous tensors. This error was linked to the saving process of the `encoder.block.0.layer.0.SelfAttention.q.weight` tensor, where the data was not in a format compatible with standard saving

procedures. The issue was resolved by creating a custom `Seq2SeqTrainer` to properly handle non-contiguous tensor data, ensuring that the model could be saved and loaded without errors. This custom trainer allowed for smooth handling of the saving process across two sets of training epochs.

- **Training Time and Resources**
  Fine-tuning the T5 model required substantial computational resources due to the large size of the model and the number of parameters and the Nvidia GTX 1050 ti card that was initially used took days (sometimes weeks) to finish training the model which needed to be monitored constantly and even then sometimes the results were very satisfying. These difficulties were also enhanced by the large volume of the AGS-Dataset which reduced to 50% for training this model, and the use of the T5-base model that is fairly more resource hungry with it 220 Million parameters compared to it's smaller version the T5-small with it's 60 Million parameters, but the use of the Nvidia L4 GPU as explained previously solved all these problem.

Despite these challenges, addressing each obstacle ultimately resulted in a more robust system. The final trained model achieved strong results in Arabic text summarization, as reflected in various evaluation metrics.

11. Training and Performance results :

The fine-tuning process for the model was divided into two phases, each consisting of 10 epochs. The objective was to observe how the model improved over time, focusing on both training loss and validation loss.

- **First 10 Epochs :**
  The first phase of training used a maximum input length of 300 tokens and

a maximum target length of 100 tokens. Below is a summary of the training and validation losses recorded at various steps:

Steps	Training Loss	Validation Loss
500	2.184000	1.612296
1000	1.906400	1.485521
…..	…..	…..
35000	1.203300	1.069621

At the end of the first 10 epochs, the training loss dropped from 2.184000 to 1.548600, and the validation loss decreased from 1.612296 to 1.309377, indicating consistent improvement.

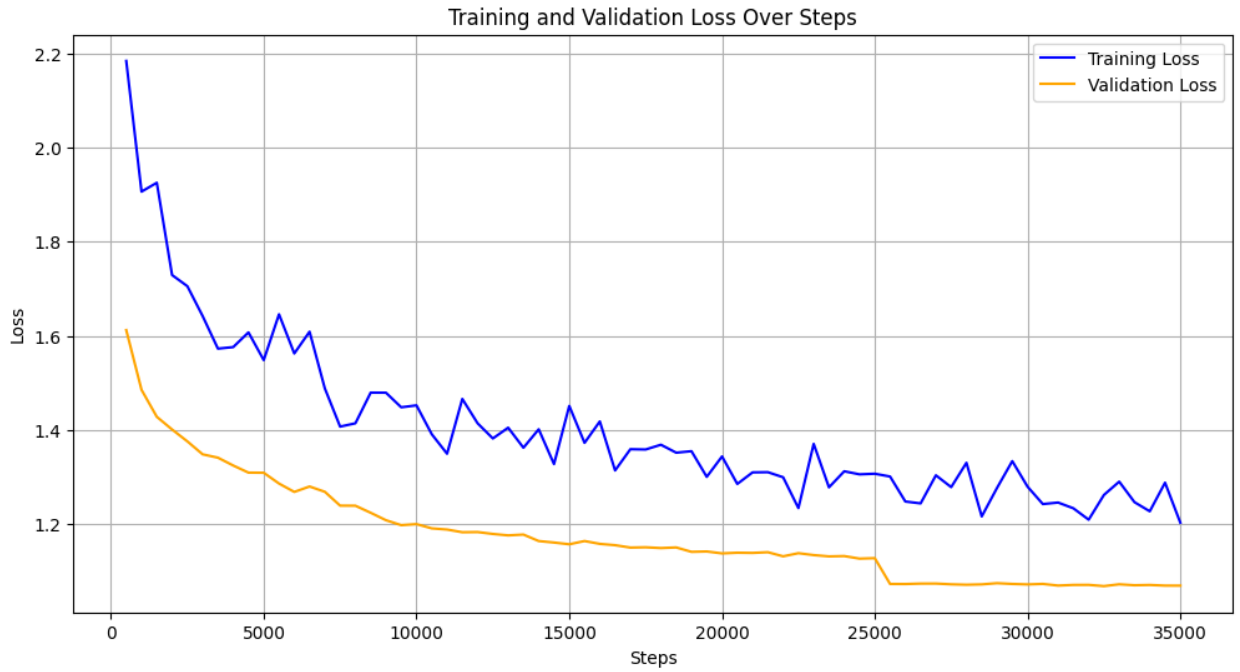

Fig 8: graph representing the training loss and validation loss over steps of the first phase.

- **Second 10 Epochs :**
  In the second phase, the maximum input length was increased to 400 tokens, and the maximum target length was set to 125 tokens. This extended dataset allowed the model to handle more complex sequences. The following table summarizes the loss values:

Steps	Training Loss	Validation Loss
500	1.143000	1.024215
1000	1.202000	1.018081
…..	…..	…..
70500	1.160100	0.952312

By the end of this second phase, the training loss reached 1.341500, and the validation loss dropped to 1.003155. This reduction in validation loss demonstrates how the model further optimized its generalization capabilities.

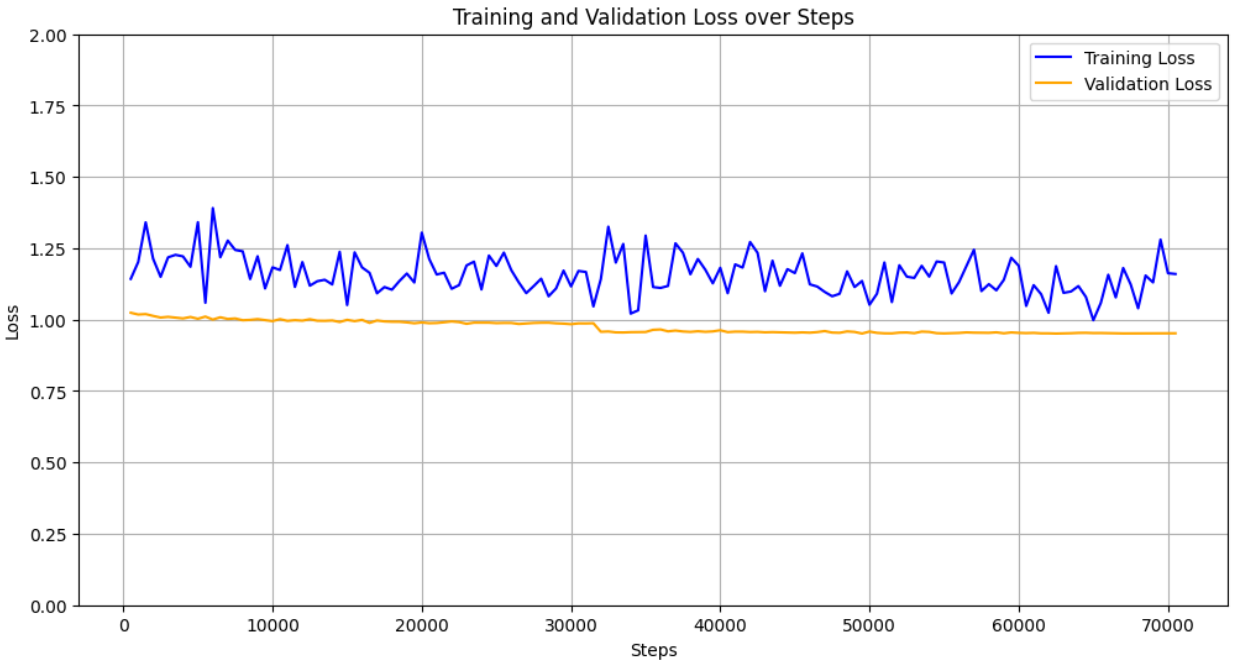

Fig 9: graph representing the training loss and validation loss over steps of the second phase.

After completing the 20 epochs, the fine-tuned model was evaluated using two key performance metrics: **BERTScore** and **ROUGE** scores, which are widely used in text summarization tasks to measure the similarity between the predicted and reference summaries.

12. Evaluation of the Model

BERTScore: For each token in the generated summary and the reference summary, embeddings are extracted from a pre-trained BERT model. These embeddings capture semantic meaning, allowing for more nuanced comparisons than word matching.

- Let $\mathbf{T_1}$ be the token embeddings of the generated (predicted) summary.
- Let $\mathbf{T_2}$ be the token embeddings of the reference summary.

The similarity between tokens is calculated using the cosine similarity between these embeddings:

$$\text{Cosine Similarity}(T_{1i}, T_{2j}) = \frac{T_{1i} \cdot T_{2j}}{\|T_{1i}\| \|T_{2j}\|}$$

Where $\mathbf{T_{1i}}$ and $\mathbf{T_{2j}}$ are the BERT embeddings of token ***i*** in the generated summary and token ***j*** in the reference summary, respectively.

- **BERTScore Precision:** The Precision, which measures how many tokens in the generated summary match the reference summary based on the embedding similarity, was **0.7658**, this indicates that most of the content in the generated summary is relevant or matches well with the reference.

  The BERTScore Precision is calculated with this Formula :

$$P = \frac{1}{|T_1|} \sum_{t_{1i} \in T_1} \max_{t_{2j} \in T_2} \text{Cosine Similarity}(t_{1i}, t_{2j})$$

- **BERTScore Recall:** The Recall which measures how well the generated summary covers the content of the reference summary, was **0.7561** a significant portion of the reference content is captured in the generated summary

  The BERTScore Recall is calculated with this formula :

$$R = \frac{1}{|T_2|} \sum_{t_{2j} \in T_2} \max_{t_{1i} \in T_1} \text{Cosine Similarity}(t_{2j}, t_{1i})$$

BERTScore F1: The F1 score, which measures the harmonic mean between precision and recall, was **0.7609**. This reflects how well the generated summaries align with the human references in terms of semantic meaning.

The BERTScore F1 is calculated with this Formula :

$$BERTScore_{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

ROUGE Scores: The ROUGE-1 score measures the overlap of unigrams (single words) between the generated and reference summaries. A score of 0.3922 indicates that 39.22% of the unigrams in the generated summaries matched with the reference summaries.

- **ROUGE-1**: 0.3922
  This indicates that 39.22% of the unigrams (words) in the generated summaries matched with the reference summaries.
- **ROUGE-2**: 0.1982
  This shows that 19.82% of the bigrams (word pairs) in the generated summaries were present in the reference summaries.
- **ROUGE-L**: 0.3725
  The ROUGE-L score, which focuses on the longest common subsequence between the generated and reference summaries, was 37.25%. This suggests that the model effectively captured the structure and context of the reference text.

The ROUGE-N Score is calculated with these formulas :

$$ROUGE - N = \frac{\sum_{\text{gram} \in \text{Reference}} \text{Count}_{\text{match}}(\text{gram})}{\sum_{\text{gram} \in \text{Reference}} \text{Count}(\text{gram})}$$

Where:

- $\text{Count}_{\text{match}}$(gram) is the number of overlapping n-grams between the generated summary and the reference summary.
- Count(gram) is the total number of n-grams in the reference summary.

And the ROUGE-L Score is Calculated with this formula :

$$ROUGE - L = \frac{LCS(\text{Generated}, \text{Reference})}{\text{Length of Reference}}$$

Where:

- LCS(Generated,Reference) is the length of the longest common subsequence between the generated and reference summaries.

ROUGE-1 evaluates unigram (word-level) overlap, **ROUGE-2** looks at bigram (two-word) overlap, and **ROUGE-L** captures sentence structure and fluency by evaluating the longest subsequence of matching words.

These performance metrics indicate that the fine-tuned model performed well in generating high-quality summaries, especially for a task as complex as Arabic text summarization.

This summary covers both the training results and the performance metrics achieved by the fine-tuned model.

13. The “clean_summary” function use:

The `clean_summary` function is designed to enhance the readability of generated summaries. It works by detecting sentence boundaries using punctuation marks with the regular expression pattern `r'[^.!?]*[.!?]'` that is designed to capture any string of characters that ends with a sentence terminator, such as a period (`.`), exclamation mark (`!`), or question mark (`?`). The function ensures that the final summary contains only complete sentences, discarding any incomplete sentence fragments at the end. This step improves the coherence and clarity of the summarized output before it is displayed to the user.

14. Comparison with Other Models

Chapter 2: Our Contribution

To evaluate the performance of our fine-tuned model for Arabic text summarization, we compared its ROUGE and BERTScore metrics against several other widely-used models in this task, specifically those trained on Arabic datasets. The following table summarizes the results:

Model	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
Our Model	39.21	19.82	37.25	76.09
eslamxm/AraT5v2-base-1024-finetuned-ar-wikilingua	26.54	10.4	23.72	72.52
ahmeddbahaa/mt5-base-finetuned-wikilingua-ar	20.79	7.60	18.81	70.87
ahmeddbahaa/mt5-base-finetune-ar-xlsum	22.20	9.57	20.26	71.43
eslamxm/mbart-finetune-ar-xlsum	15.56	4.64	13.59	71.53

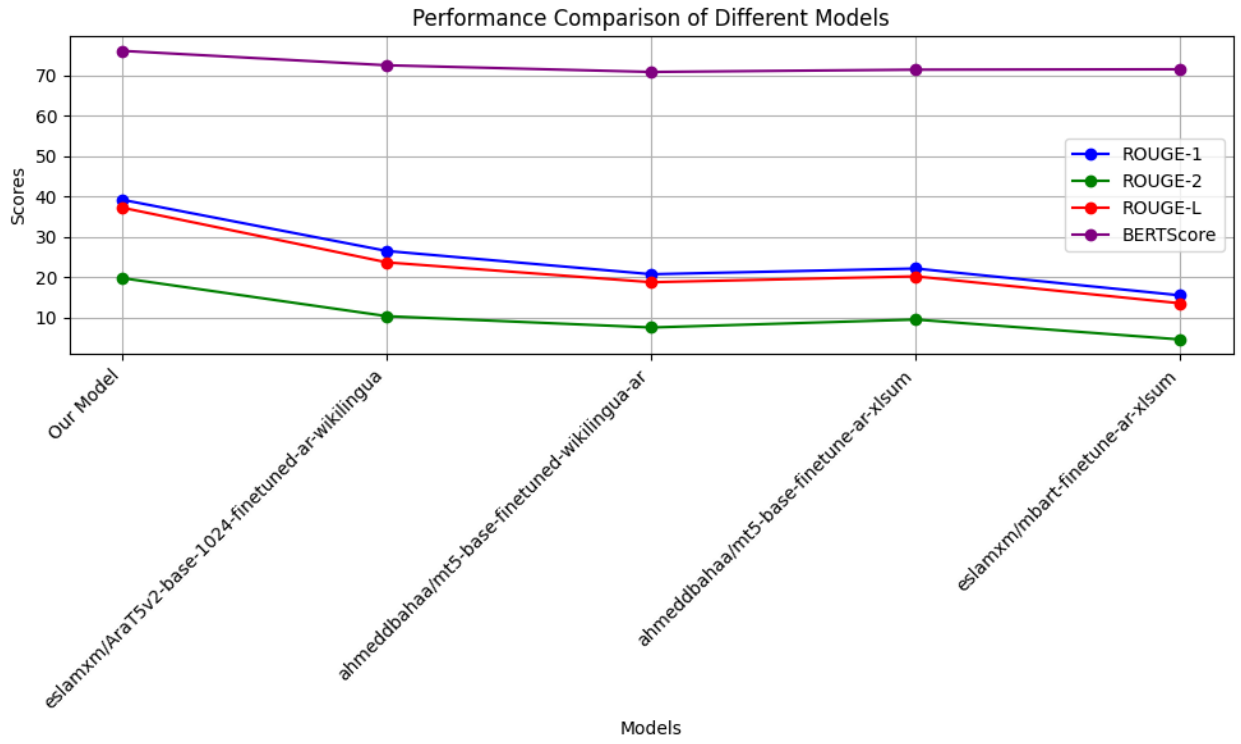

Fig 10: A graph comparing the performance of different models across the ROUGE-1, ROUGE-2, ROUGE-L, and BERTScore metrics.

1. **Performance Analysis**
   - **Our Model** demonstrates a significant improvement across all metrics compared to the other models, achieving the highest **ROUGE-1** (39.21%), **ROUGE-2** (19.82%), **ROUGE-L** (37.25%), and **BERTScore** (76.09%).
   - The **eslamxm/AraT5v2-base-1024-finetuned-ar-wikilingua** model comes in second, but its performance (ROUGE-1: 26.54%) is noticeably lower compared to our model, particularly in **ROUGE-2** and **ROUGE-L**, which measure the generation of more complex word sequences and structure.
   - Models like **ahmeddbahaa/mt5-base-finetune-ar-xlsum** and **eslamxm/mbart-finetune-ar-xlsum** lag behind, particularly in bigram generation (**ROUGE-2**) and sentence structure accuracy (**ROUGE-L**).

Our model's higher **BERTScore** of 76.09% also indicates superior semantic accuracy, ensuring that the generated summaries align better with human reference summaries than the compared models.

This comparison highlights the effectiveness of the fine-tuning techniques and pre-processing steps used in our model, making it a top-performing solution in Arabic text summarization.

15. Application Deployment

The Arabic text summarization application was successfully deployed on the Streamlit platform, which provides an easy-to-use interface for web app development and hosting. Streamlit was chosen due to its compatibility with Python-based applications, allowing for seamless integration with the fine-tuned T5 model used in the project.

Deployment Process

The deployment involved uploading the necessary code, along with the model and tokenizer, to the Streamlit cloud infrastructure. The front-end interface was developed using Streamlit's framework, making it possible for users to interact with the application directly through a browser.

The application is accessible on the web, where users can input Arabic text, trigger the summarization process, and receive a cleaned summary output in real-time. This deployment strategy ensures that the app is user-friendly and accessible, without requiring local installation or complex setup.

By hosting the app on Streamlit servers, maintenance and updates are also simplified, allowing for quick iterations on both the user interface and backend processes.

16. Application Testing

To ensure the functionality and usability of the Arabic text summarization application, a comprehensive testing phase was conducted. The following images illustrate the user journey through the application.

- **Home Screen**

Upon launching the application, users are greeted with a clean and simple interface, where they are invited to input Arabic text that they wish to summarize. The home screen provides a text input area and a "Summarize" button.

Fig 11: A screenshot showing the home screen of the app.

- **Text Input**

In this stage, users copy or type their Arabic text into the input field. Once the text is entered, they can click the "Summarize" button to initiate the summarization process. The following image shows the application after the text has been entered.

Chapter 2: Our Contribution

Fig 12: A screenshot showing the text being imputed and ready for summarization.

- **Summary Generated**

After processing, the application generates a cleaned summary of the input text. The summary is displayed in a separate section, showcasing the system's ability to distill the input into a concise, readable format.

This testing phase ensured that the application correctly handles various text inputs, processes them efficiently, and provides accurate summaries. The deployment of the app on Streamlit servers allowed for real-time user interaction during testing.

Chapter 2: Our Contribution

Fig 13: An image shows the summary of the input text being generated after the user clicks on the “تلخيص” button.

- **Exceptions handling :** If the user leaves the input area blank or enters a non- Arabic text, a warning message in red appears telling the user to enter an Arabic text and not leave the input area empty.

Fig 14: An image shows the warning message that the user receives if he clicks on the "تلخيص" button while the input area is blank.

Fig 15: An image shows the warning message that the user receives if he enters a none-arabic text to summarize and clicks on the "تلخيص" button.

17. Conclusion

In this chapter, we have detailed the development and implementation of our Arabic text summarization system. Starting with the selection of the AGS-Corpus dataset and utilizing Google's T5-base model and tokenizer, we pre-processed the data to optimize it for training. The fine-tuning process was carried out over 20 epochs, overcoming challenges related to tensor saving and incomplete summary generation through the implementation of custom functions, such as the `clean_summary` function.

We evaluated the model using key performance metrics, including BERTScore and ROUGE, achieving competitive scores that outperform several established models in the field. Additionally, we discussed the deployment of the application on Streamlit servers, showcasing its practical utility as a user-friendly web app for Arabic text summarization.

Through these contributions, our work demonstrates the potential of transformer-based models for Arabic text summarization while addressing specific challenges related to text generation quality and deployment in real-world applications.

General Conclusion

1. Restating Research Objectives

The overarching aim of this research was to develop an advanced, efficient, and accurate Arabic text summarization system, capable of meeting the unique challenges posed by the Arabic language. Arabic, with its rich morphology, diverse dialects, and syntactic complexity, presents significant difficulties in the realm of Natural Language Processing (NLP). These linguistic features, including root-based word formation, complex grammatical rules, and a wide range of word variations, necessitate specialized approaches to achieve high-quality text summarization. Our objective was to address these challenges by employing state-of-the-art deep learning methodologies, particularly transformer-based models, to optimize the summarization process for both Modern Standard Arabic (MSA) and potentially for dialectal variants as well.

The initial goal was to leverage existing pre-trained models, such as Google's T5-base, and fine-tune them for Arabic text, which required customizing various stages of data pre-processing, tokenization, and model optimization to suit the language’s structural and contextual requirements. The ultimate aim was not just to develop a functional summarization system, but to create one that excels in accuracy and fluency, producing summaries that closely align with the meaning, context, and style of the original Arabic texts.

In addition to developing the model, we also sought to make the summarization system accessible to a broader audience. This involved integrating the fine-tuned model into a user-friendly, web-based application, making it easy for non-technical users to generate summaries without requiring deep knowledge of NLP or machine learning. The deployment of this system aimed to create practical, real-world utility, offering Arabic speakers an efficient tool for text summarization in both academic and everyday contexts.

The objectives also included evaluating the model’s performance against established benchmarks in the field of Arabic text summarization. To ensure that our system provided meaningful improvements, we set out to compare it with other state-of-the-art models, using key metrics like BERTScore and ROUGE.

General Conclusion

This comprehensive evaluation aimed to not only validate the model's capabilities but also to highlight areas for future research and improvements in Arabic NLP.

2. Summary of Contributions

- **Development of a Fine-Tuned Model:** We fine-tuned Google's T5-base model on the AGS-Corpus dataset, customizing the tokenization and pre-processing steps to better suit Arabic text summarization.
- **Performance Evaluation:** The model was evaluated against widely recognized metrics (BERTScore and ROUGE) and demonstrated significant improvements over other existing Arabic summarization models, outperforming them in terms of both unigram and bigram matching.
- **Web Application Deployment** A web-based interface was developed using Streamlit, offering users an intuitive platform to input Arabic text and generate clean, accurate summaries.
- **Comparison with Existing Models:** Our model was compared with other state-of-the-art models, demonstrating superior performance across most metrics, especially in ROUGE-1 and BERTScore.

3. Key Findings

Our work has demonstrated that a fine-tuned T5-base model can significantly improve the quality of Arabic text summarization, especially in terms of capturing both lexical and semantic accuracy. The deployment of the model as a web application further highlights its practical value, offering an easy-to-use tool for Arabic speakers. Our system's higher ROUGE and BERTScores compared to other models confirm its robustness and effectiveness.

4. Challenges and Limitations

- **Data Limitations:** Despite using the AGS-Corpus, the scarcity of high-quality Arabic text summarization datasets posed a challenge, limiting the scope of the model's training.
- **Model Truncation Issues:** Truncation errors caused by token limits in the model were mitigated by using the clean_summary function, though this only partially alleviated the issue.

- **Deployment Constraints:** While deploying the web application, certain limitations of the Streamlit platform were encountered, particularly related to processing large input texts in real-time.

5. Future Work

- **Dataset Expansion:** Increasing the dataset size and diversity, including more varied Arabic dialects, would likely improve the model's generalization capabilities.
- **Improved Pre-processing Techniques:** Experimenting with more advanced tokenization strategies and addressing truncation problems could further enhance summarization accuracy.
- **Incorporating Transformer Variants:** Testing with other transformer-based models (like BART or GPT variants) might lead to even better performance for Arabic text summarization.
- **User Feedback Integration:** Incorporating user feedback within the deployed app could help in fine-tuning the model further based on real-world usage patterns.

References

[1] El-Kassas, Wafaa & Salama, Cherif & Rafea, Ahmed & Mohamed, Hoda. (2020). Automatic Text Summarization: A Comprehensive Survey. Expert Systems with Applications. 165. 113679. 10.1016/j.eswa.2020.113679.

[2] Adhika Pramita Widyassari, Supriadi Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, Affandy Affandy, De Rosal Ignatius Moses Setiadi,
Review of automatic text summarization techniques & methods,
Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 4, (2022).

[3] Al-Maleh, M., Desouki, S. Arabic text summarization using deep learning approach. *J Big Data* 7, 109 (2020).

[4] A. Elsaid, A. Mohammed, L. F. Ibrahim and M. M. Sakre, "A Comprehensive Review of Arabic Text Summarization," in *IEEE Access*, vol. 10, pp. 38012-38030, 2022, doi: 10.1109/ACCESS.2022.3163292. keywords: {Deep learning;Standards;Measurement;Computer science;Data mining;Computer architecture;Complexity theory;Text summarization;arabic natural language processing;machine learning;extractive text summarization;abstractive text summarization;and deep learning models}.

[5] Antoun, Wissam & Baly, Fady & Hajj, Hazem. (2020). AraBERT: Transformer-based Model for Arabic Language Understanding.

[6] Farghaly, Ali & Farghaly, Ali & Shaalan, Khaled & Khaled,. (2009). Arabic Natural Language Processing: Challenges and Solutions. ACM Transactions on Asian Language Information Processing (TALIP). 8. 1-.

[7] Shaalan, Khaled. (2010). Nizar Y. Habash, Introduction to Arabic natural language processing (Synthesis lectures on human language

technologies). Machine Translation. 24. 285-289. 10.1007/s10590-011-9087-8.

[8] Abdelali, Ahmed & Mubarak, Hamdy & Samih, Younes & Hassan, Sabit & Darwish, Kareem. (2020). Arabic Dialect Identification in the Wild. 10.48550/arXiv.2005.06557.

[9] Abdelqader, Khaled & Mohamed, Azza & Shaalan, Khaled. (2023). Systematic Review of Automatic Arabic Text Summarization Techniques. 10.1007/978-981-99-3416-4_63.

Webography

FIG 1 : Extractive summarization extracts key terms verbatim. Abstractive summarization paraphrases the document. https://aws.amazon.com/blogs/media/democratize-documentation-summarization-with-hugging-face-on-amazon-sagemaker/ **[first accessed 21 August 2024]**

FIG 2 : Meta-Heuristic Algorithms-Tuned Elman vs. Jordan Recurrent Neural Networks for Modeling of Electron Beam Welding Process - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Flowchart-of-a-recurrent-neural-networks-tuned-by-an-optimization-algorithm_fig3_350042261 **[first accessed 21 August 2024]**

FIG 3 : Short-Term Traffic Prediction Using Deep Learning Long Short-Term Memory: Taxonomy, Applications, Challenges, and Future Trends - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Flow-chart-of-LSTMs-operation-The-yellow-pink-and-green-blocks-indicate-the-forget_fig4_373505006 **[first accessed 22 August 2024]**

FIG 4 : A Deep Convolutional Neural Network to Analyze Position Averaged Convergent Beam Electron Diffraction Patterns - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Flow-chart-of-the-convolutional-neural-networks-implemented-during-the-automated_fig2_318899360 **[First accessed 22 August 2024]**

FIG 5 : Modeling monthly rainfall of India through transformer-based deep learning architecture - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Flowchart-of-proposed-Transformer-

based-deep-learning-algorithm_fig5_378074251 [First accessed 22 August 2024]

FIG 6 : Comparative Study of Arabic Stemming Algorithms for Topic Identification - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Example-of-an-Arabic-word-morphology-analyzed_fig1_336542891 [First accessed 22 August 2024]

FIG 7 : Abstractive Text Summarisation using T5 Transformer Architecture with analysis - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Text-to-text-Transfer-Transformer-T5-Model_fig2_383526154 [First accessed 22 August 2024]

Abstract

This thesis presents an advanced Arabic text summarization system built by fine-tuning a T5-base transformer model on the AGS-Corpus, a curated Arabic summarization dataset. The model effectively handles the language's rich morphology, diverse dialects, and complex syntax. Evaluation using BERTScore and ROUGE shows that the system generates high-quality summaries closely aligned with human references in content, fluency, and semantic similarity. To improve accessibility, the system is deployed as a user-friendly web application using Streamlit, enabling real-time summarization for non-technical users. Future work includes expanding the dataset to include more dialects and exploring alternative architectures like BART or GPT to further enhance performance, especially on complex or lengthy texts.

Résumé

Cette thèse propose un système avancé de résumé automatique en arabe, basé sur un modèle T5-base affiné sur le corpus AGS. Grâce à cette approche, le système surmonte les défis liés à la morphologie, aux dialectes et à la syntaxe de la langue arabe. Les résultats, évalués par BERTScore et ROUGE, démontrent une haute qualité des résumés générés, proches de ceux produits par des humains en termes de contenu et de fluidité. Le modèle est intégré dans une application web développée avec Streamlit, offrant une interface intuitive pour un usage en temps réel par des utilisateurs non spécialistes. Les perspectives incluent l'enrichissement du corpus et l'expérimentation de modèles comme BART ou GPT pour mieux traiter les textes longs et complexes.

عربي

تقدّم هذه الأطروحة نظامًا متقدمًا لتلخيص النصوص العربية، يعتمد على تحسين نموذج T5-base مدرّب على مجموعة بيانات AGS. أظهرت يعالج النموذج بنجاح تعقيدات اللغة العربية من صرف ولهجات وبنية نحوية. التقييمات باستخدام ROUGE وBERTScore أن النظام ينتج تلخيصات عالية الجودة ومتشابهة مع التلخيصات البشرية من حيث المحتوى والسلاسة. تم نشر النموذج كتطبيق ويب باستخدام Streamlit لتسهيل استخدامه من قبل غير المتخصصين. يشمل العمل المستقبلي توسيع قاعدة البيانات لتشمل لهجات إضافية وتجريب نماذج بديلة مثل GPT وBART لتحسين الأداء مع النصوص المعقدة والطويلة.