Comparing the Effectiveness of Stemming and Lemmatization in Arabic Text Summarization with XLSum

Team 5: Ahmed Eid, Marwan Farag, Mohamed Abdelhamid

May 24, 2023

Abstract

Text summarization is a challenging task that aims to produce a concise summary of a given text while retaining its most important information. In Arabic, this task is particularly challenging due to the complex morphology and rich lexical diversity of the language. Stemming and lemmatization are common techniques used in NLP to reduce the dimensionality of text data and improve text summarization. In this paper, we investigate the impact of stemming and lemmatization on Arabic text summarization with the XLSum dataset. We compare the effectiveness of the two techniques in improving the performance of summarization models and evaluate their impact on the quality of the generated summaries. We also explore the impact of not using stemming or lemmatization on text summarization.

The results show that preprocessing without stemming or lemmatization produces the best outcomes for Arabic automatic text summarization. Moreover, although computationally expensive, lemmatization performs better than stemming. These findings expand NLP research in Arabic language and offer insightful suggestions for enhancing Arabic text summarization methods.

1 Introduction

A challenging task in natural language processing (NLP) is automatic text summarization, which tries to produce a shortened version of a given text while keeping its key information. Text summarization is a difficult process in Arabic NLP due to the complexity and richness of the Arabic language as well as its wide-ranging and enormous vocabulary. In addition to the language's complexity, there are not many datasets for high-quality Arabic text summarization.

1.1 Motivation

As stated, Arabic text summarization has become an increasingly important task due to the massive amount of Arabic text available on the internet and in various fields. Compared to Arabic, there is a lot more NLP research on Latin languages, notably summarization.

There are several applications of Arabic text summarization. For example, news aggregation is a significant application. Text summarization can be used to aggregate news articles from various sources and provide users with a summary of the most important events and developments [GSK18]. This can save time and help users stay informed. In addition, text summarization can be useful in legal documents [JBB21]. Legal documents can be long and complex, and summarization can be used to extract key information, making it easier for lawyers and other legal professionals to review and understand the document. Moreover, Summarization can be used to extract key insights and trends from business reports [PBS23], allowing decision-makers to quickly identify important information and make informed decisions. Lastly, text summarization can be used in educational resources [SBS14]. It can be used to create study materials, summaries of textbooks, and other educational resources, helping students to better understand complex concepts and ideas. In conclusion, Arabic text summarization is a useful tool that can be used in a variety of fields to save time, provide better accessibility, and extract key information.

The challenge of deciding whether to use stemming or lemmatization is a common issue in natural language processing [KSND21], and it can be particularly challenging for Arabic text. While stemming reduces words to their root form by removing affixes, lemmatization maps words to their base or

dictionary form while retaining the meaning of the word. The choice between stemming and lemmatization depends on the task at hand and the specific characteristics of the language being analyzed. By intuition, we were concerned that stemming is very strict in the sense that returns words to their roots (ex: استثمر changed to ثمر which somehow has different meaning). At the same time stemming is appropriate identifying common roots. Arabic has a rich system of word roots, with many words sharing the same root. For example, the root كتب (k-t-b) is the basis for words like كاتب (kātib) meaning "writer", كتابة (kitābah) meaning "writing", كتاب (kitāb) meaning "book", and many others. Hence, Stemming algorithms can group together words that share a common root, making it easier to identify key themes in a text and create a summary that captures the essence of the original content. To the best of our knowledge, there has been no studies that try to compare between the two approaches for preprocessing. In [EBEK20], they claim that although lemmatization is more time-expensive, doing it preprocessing step will give the best accuracy. Other researches, however, insist on using stemming as a preprocessing step [BGE+20].

1.2 Related Work

This section will discuss some of the other approaches that might be used to achieve the task of automatic text summarization. Extractive summarization methods based on statistical [AFA12] and graph-based [ATAO14] approaches have been widely used due to their simplicity and effectiveness. However, they have several limitations, including the lack of contextual understanding, difficulty in handling ambiguity, vulnerability to noise, and lack of coherence. On the other hand, abstractive summarization methods, such as tree-based summarization [KM00] and deep learning Seq2Seq models, have shown promising results in overcoming these limitations. In particular, AraBART, a deep learning model specifically designed for Arabic text summarization, has achieved state-of-the-art performance on multiple datasets, outperforming strong baselines such as pretrained Arabic BERT-based models and multilingual mBART and mT5 models [ETH+22]. Other methodologies, such as semantic-based summarization and ontology-based summarization, also hold potential for further exploration in Arabic text summarization research.

1.3 XLSum Dataset

The XLSum Arabic dataset [HBI+21] is a large-scale Arabic text summarization dataset that was released in 2020. It contains a diverse collection of news articles and their corresponding summaries in Arabic. The dataset was created by collecting news articles from various Arabic news sources and then manually creating summaries for each article. The dataset is freely available for research purposes and has received significant attention from the NLP community as a valuable resource for Arabic text summarization research.

The XLSum Arabic dataset consists of approximately 45,000 news articles collected from various Arabic news sources, such as Al Jazeera, Al Arabiya, and BBC Arabic. The articles cover a wide range of topics, including politics, economics, sports, and culture. Each article in the dataset is accompanied by a corresponding summary, which was manually created by a human summarizer. The summaries are one sentence or a few sentences long and aim to capture the most important information in the article.

The dataset is split into three parts: a training set, a validation set, and a test set. The training set contains approximately 35,000 articles, while the validation and test sets contain approximately 5,000 articles each. The dataset is designed to be used for supervised learning tasks, such as training and evaluating text summarization models.

1.4 Stemming and Lemmatization

Stemming and lemmatization are common techniques used in NLP to reduce the dimensionality of text data and improve text summarization. Stemming involves reducing words to their root form. It is one of the methods employed by information retrieval systems to ensure that word variants are not missed when text is retrieved [BLY14]. Lemmatization, on the other hand, involves reducing words to their base form, typically by using a dictionary or morphological analysis. it aids in matching synonyms while using a thesaurus, thus when one searches for "hot," the term "warm" is also matched [BLY14].

Stemming and lemmatization are widely used in text processing tasks, including text classification [TTJ06], sentiment analysis [BQWR14], and text summarization [TM12].

In this paper, we investigate the impact of stemming and lemmatization on Arabic text summarization with the XLSum dataset. We compare the effectiveness of the two techniques in improving the performance of summarization models and evaluate their impact on the quality of the generated summaries. We also explore the impact of not using stemming or lemmatization on text summarization. Our goal is to provide insights into the usefulness of these techniques in Arabic text summarization and identify the most effective approach for this task and to how extent it is effective compared to the others.

2 Dataset Challenges

2.1 Limited Genre Coverage

The XLSum dataset might only comprise a small amount of data from a given genre or domain. It mainly concentrates on particular subjects or styles of documents, which might restrict how generalizable it is to other text summary tasks. The dataset is mostly made up of reports, which might not sufficiently cover the language and idioms used in everyday situations or casual settings. The delivery of factual information in news articles frequently involves a particular journalistic style, which can lead to the use of more formal and constrained language. People use many different words, idioms, slang, and colloquial expressions in everyday speech that might not be well-represented in the dataset. As a result, text summarization models may not be exposed to the entire spectrum of linguistic nuances present in casual written texts or everyday conversations, which might provide problems when training them.

2.2 Limited Human Annotation

The XLSum dataset's summaries were created by a small group of human annotators, which raises the possibility of subjectivity and errors. Due to the fact that individual annotators may have their own interpretations and standards when choosing and presenting data, these restrictions may affect the dataset's dependability and quality. This can impact the consistency and objectivity of the dataset and lead to differences in the substance of the summaries. The dataset's minimal human annotation raises additional potential difficulties due to the absence of inter-annotator agreement and the increased chance of errors.

2.3 Limited Text Length Variation

The inadequate representation of text length variation often found in real-world contexts is one drawback of the XLSum dataset for text summarising. Lack of a wide variety of document lengths in the dataset may affect the model's capacity to handle longer texts or efficiently produce brief summaries for shorter texts. Text documents' lengths can differ greatly in real-world contexts. Other documents might be brief news snippets, social media posts, or product descriptions, while some might be extensive articles, reports, or research papers. A dataset can help equip models to handle the inherent difficulties involved in summarising texts of various sizes by include a wide range of text lengths.

Models trained on the XLSum dataset may have trouble summarizing larger documents when there is insufficient text length variance in the dataset. It might be difficult for models to concisely summarise the most important ideas in longer papers because they frequently contain more extensive information and complicated structures. The generation of coherent and illuminating summaries for such texts may prove challenging for models trained on datasets with limited representation of long documents.

2.4 Dataset Size and Training Time

The Xlsum dataset is quite large, which can result in significant training time requirements. This posed a challenge for us, since we have limited computational resources and time constraints. Therefore, we needed to only train our model on only 20% of its size. The need to reduce the dataset size to address this limitation indicates that the original dataset may not be easily manageable for certain resources.

3 Methodology

3.1 Preprocessing Steps

3.1.1 Removing Homographic Words

One challenge is dealing with homographic words, which are words that are written differently but have the same meaning. Arabic has several homographic words, such as the use of ه and ة at the end of words, or ي and ى. For example, the two words المستقبلية and المستقبليه may both appear in the dataset, represented by two different words although they should be the same word. Homographic words can create confusion and lead to errors in text summarization. It is important to identify and deal with homographic words to ensure that the summarization is accurate. The severity of this challenge is high because homographic words can cause significant errors in the summarization process. From analysis of the dataset, it has been found that around 5600 words appear in different shapes.
Therefore, we decided to normalize by replacing ة with ه and ى with ي. This will somehow help in normalizing same words that have different shapes to the same shape to avoid confusing the model.

3.1.2 Removing Stop Words

By removing stop words, we can improve the accuracy of the topic modeling algorithm, extract more meaningful topics and reduce the size of the vocabulary to make it easier to analyze and process the text.

3.1.3 Diacritics and punctuation

Diacritics are small marks used in Arabic to indicate pronunciation, but they are not essential for understanding the text. Removing them can simplify text comparison for summarization, but it may also increase ambiguity. Hence, diacritics and punctuation are removed from input articles and their summaries.

3.2 System Architecture

A model for text summarization should typically consist of several layers that perform different functions to process the input text and generate a summary output. Here are some common layers that we thought are crucial for a text summarization model:

- An Embedding layer: we need to map the input tokens into word embeddings that the model can understand.
- An Encoder: this encoder should consist of multiple multi-head attention layers to make use of the attention mechanism and positional encoding to capture the context of the words.
- Feed-forward layer: this layer should help the model extract features and learn complex non linear relations.
- Decoder layer: the decoder layer generates the summary text by autoregressively predicting each word in the summary sequence based on the previous words it has generated. This layer typically consists of multiple stacked transformer blocks, each of which includes multi-head self-attention, cross-attention, and feedforward layers. The cross-attention mechanism allows the model to attend to the relevant parts of the input text while generating the summary.
- Output Layer: This layer produces the final summary text. This layer typically uses a softmax activation function to compute the probabilities of each word in the summary vocabulary, and the word with the highest probability is selected as the next word in the summary sequence.

AraBART [ETH+22] is a useful pretrained model that we can use and fine-tune to the summarization task as it already has the above needed layers (See Figure 1). Each layer in AraBART is useful for text summarization in different ways:

1. Tokenization layer: AraBART uses a custom Arabic-specific tokenization scheme to tokenize the input text into subwords. This is necessary to handle the rich morphology of Arabic words, which can have multiple affixes and inflections.

2. Embedding layer: The embedding layer maps each subword token in the input sequence to a high-dimensional vector space, where each dimension represents a different feature or aspect of the token. AraBART uses a shared vocabulary for both input and output sequences, which allows the model to share the embeddings between the encoder and decoder.

3. Encoder layer: The encoder layer processes the input sequence and produces a condensed representation of the input text. The encoder in AraBART consists of 12 transformer blocks, each of which includes multi-head self-attention and feedforward layers. The encoder also uses position embeddings to preserve the order of the input sequence.

4. Decoder layer: The decoder layer generates the summary text by autoregressively predicting each subword token in the summary sequence based on the previous subwords it has generated. The decoder in AraBART consists of 12 transformer blocks, each of which includes multi-head self-attention, cross-attention, and feedforward layers. The decoder also uses position embeddings and a mask to ensure that each token is generated based only on the previous tokens.

5. Output layer: The output layer produces the final summary text by selecting the subword token with the highest probability in the summary vocabulary. The output layer in AraBART uses a linear layer followed by a softmax activation function to compute the probabilities of each token in the summary vocabulary.

3.3 Training Details

The AraBART model served as the pretrained model during the training procedure. The corresponding pretrained tokenizer was loaded to confirm that the model architecture and tokenization were compatible. As a result, the tokenizer was guaranteed to match the particular model being used. To prepare the data for training, the full dataset was tokenized using the pretrained tokenizer. This made it possible to separate the text into single tokens, which serve as the model's fundamental input units. The pretrained model was then adjusted using PyTorch's Trainer class. The training process was made simpler by the Trainer class, which offered a feature-rich training API for a variety of common use cases. Batches of data for training were produced using a unique type of data collator. Data collators are objects that produce a batch of data from a list of dataset elements as input. To match the maximum length of each batch, the DataCollatorForSeq2Seq was used in this instance. It dynamically

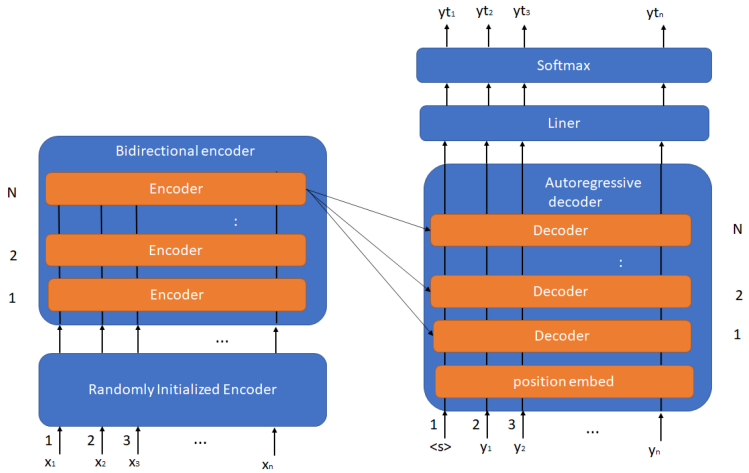

Figure 1: This figure shows the model architecture of [LLG+19] as presented by [AGN+22]. This was the basis of inspiration that [ETH+22] used to train their AraBART model and train it on Arabic datastet.

padded the inputs and labels.
The training process's learning rate was set to 5×10^{-5}. The gradient descent optimisation step size was dictated by this hyperparameter, which also had an effect on convergence speed and quality.

The model was trained using an initial set of three training epochs, or the number of times the full dataset is run through the model. Depending on the requirements and features of the dataset, this number might need to be modified. A batch size of 4 was chosen to handle memory and computing resources effectively. It included breaking up the data into batches, each of which had four samples.
The Seq2SeqTrainer was created by combining all the aforementioned elements, including the pre-trained model, data collator, datasets, and evaluation metric. The Seq2SeqTrainer class, which made use of the supplied components and carried out the required steps for efficient model training, facilitated the training process.
Please note that the training process was realized to expense to relatively high expenses in time and computatational resources. As pointed out in section 2.4, since we are limited in time and computational resources. this was one of the challenges that has been faced while training. This challenge has been mitigated by cutting out the training and dataset to the 20% of its size. This means training data comprised about $7,000$ datapoints while testing data comprised about 900 datapoints.
It is important to mention that 3 setups of training have been conducted. All the three setups have the same preprocessing steps mentioned in section 3.1. They only differed in the following:

1. Training Setup #1: Preprocessed data involved stemming.
2. Training Setup #2: Preprocessed data involved lemmatization.
3. Training Setup #3: Preprocessed data did not involve neither stemming nor lemmatization.

The above training setups will be refereed to frequently in the following sections.

3.4 Evaluation

Automatic text summarization systems are frequently evaluated using the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics. By comparing the generated summaries to the reference summaries offered in the evaluation dataset, they determine how similar they are. There are various ROUGE variations, the most popular ones being ROUGE-1, ROUGE-2, and ROUGE-L. Only the stat F1-measure for the three metrics ROUGE-1, ROUGE-2, and ROUGE-L is of importance to us. The definitions of each metric are as follows:

3.4.1 ROUGE-1 (Unigram ROUGE)

ROUGE-1 calculates the amount of overlap between the generated and reference summaries in terms of unigrams (single words). Based on the unigram matches, the precision, recall, and F1-score are calculated. The precision of a generated summary is defined as the proportion of correctly matched unigrams to the total number of unigrams in the created summary. Recall is the number of unigrams in the generated summary that were properly matched to those in the reference summary as a whole. The harmonic mean of recall and precision yields the F1-score, which offers a fair assessment of their relative trade-offs. In general, the equation for calculating the F1 score for ROUGE-1 summary can be expressed as follows:

$$F1_{\text{score\_ROUGE-1}} = \frac{2 \times (precision_{\text{ROUGE-1}} \times recall_{\text{ROUGE-1}})}{precision_{\text{ROUGE-1}} + recall_{\text{ROUGE-1}}} \tag{1}$$

where the equation for calculating the precision score for ROUGE-1 summary can be expressed as follows:

$$Precision_{\text{ROUGE-1}} = \frac{\text{Count(unigrams in common between generated summary and reference summary)}}{\text{Count(unigrams in generated summary)}} \tag{2}$$

and the equation for calculating the recall score for ROUGE-1 summary can be expressed as follows:

$$Recall_{\text{ROUGE-1}} = \frac{\text{Count(unigrams in common between generated summary and reference summary)}}{\text{Count(unigrams in reference summary)}} \tag{3}$$

3.4.2 ROUGE-2 (Bigram ROUGE)

Similar to ROUGE-1, ROUGE-2 measures the overlap of bigrams (pairs of immediately following words) as opposed to unigrams. Based on the bigram matches between the generated summary and the reference summary, it determines precision, recall, and F1-score. ROUGE-2 collects some contextual information by taking word pairs into account and can offer a more thorough assessment of the quality of the summary. Note that the equations for calculating ROUGE-2 are the same as equations 1, 2 and 4 except that bigram counts are used instead of unigram counts.

3.4.3 ROUGE-L (Longest Common Subsequence)

The generated summary and the reference summary's longest common subsequence is measured by ROUGE-L. A subsequence is a group of words that appear together but are not necessarily sequentially. The length of the longest common subsequence is used by ROUGE-L to calculate precision, recall, and F1-score. Even when there are little differences in wording or word arrangement, this metric effectively captures the structural similarities between summaries.

$$ROUGE - L = \frac{LCS(\text{generated summary}, \text{reference summary})}{\max(\text{length of generated summary}, \text{length of reference summary})} \tag{4}$$

where LCS referes to the *longest common subsequence*.
It is important to mention that our test dataset comprised 900 datapoints and their respective summaries as the labels. The three mentioned metrics are calculated for all of the test data points and then the average is taken to evaluate each of the three training setups.

4 Results and Analysis

This section will present and analyze the results of training the model on stemmed, lemmatized, or neither stemmed no lemmatized data.

Figure 2 shows the ROUGE-1 results for the three experiment setups. A high ROUGE-1 score implies a considerable unigram (individual word) overlap between the reference summary and the generated summary. It can be noticed that stemming and lemmatization do not help to effectively capture most of the key information in the reference summary. Nevertheless, we need not solely depend on

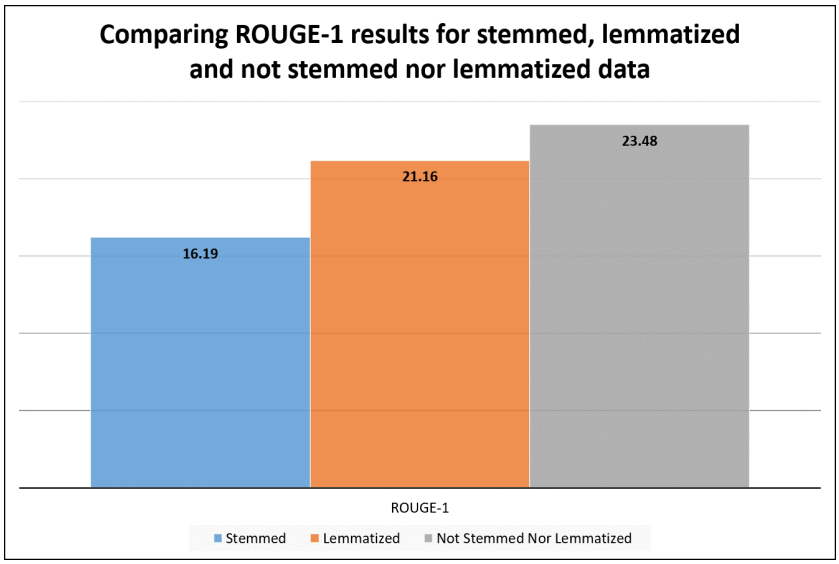

Figure 2: Comparison between training the AraBART model on stemmed data, lemmatized data and neither stemmed nor lemmatized data. The figure depicts the average Rouge-1 on the test dataset. This shows how stemming and lemmatization have negative impact on the properness of the produced summaries.

the ROUGE-1 score since ROUGE-1 is a constrained assessment metric that does not take the words' context or semantic significance into account. It only emphasises the overlap of the unigrams. The quality or coherence of the resulting summary is not always guaranteed by a high ROUGE-1 score. For a thorough review, additional evaluation techniques are frequently needed.

This is why Rouge-2 has been considered as another metric to, hopefully, capture the context. Figure 3 shows how each training setup performs on the ROUGE-2 metric. It can be noticed that, again, preprocessed data with neither stemming nor lemmatization (training setup #3) outperformed the training setups #1 and #2 with data preprocessed with stemming or lemmatization. A high ROUGE-2 score of training setup 3 indicates that the generated summary keeps the word order and context in addition to capturing specific words from the reference summary.

Finally, Figure 4 shows the results of the three training setups on the ROUGE-L metric. It can be noticed that, for the third time, training setup #3 performed better on the ROUGE-L metric. A strong overlap between the generated summary and the reference summary's longest common subsequence (LCS) is indicated by a high ROUGE-L score. The longest common subsequence, or the longest group of words that appear in the same order in both summaries, is used by the ROUGE-L metric to assess how similar the generated and reference summaries are to one another. A high ROUGE-L score indicates that the generated summary accurately captures the key information and keeps the primary organizational features of the reference summary. By aligning the generated summary with the reference summary in terms of the longest common subsequence, it shows that the summarizing algorithm properly incorporates crucial information. The results show that training setup #3, which involved preprocessing the data without any stemming or lemmatization, outperformed training setups #1 and #2, which each featured stemming and lemmatization, in terms of performance. This shows that stemming and lemmatization might be detrimental to AraBART's ability to summarise abstractive texts.
The modification of word structure before entering it into the model can be attributed for the unsuccessful results of lemmatization and stemming. For instance, lemmatization changes words like كتبوا "they wrote" into كتب "he wrote" by removing affixes. This deletion of plurality from the original word disregards crucial information that the model might need, which could cause confusion and result in summaries that are missing crucial information.

Stemming, on the other hand, eliminates plurality and other linguistic features, which poses prob-

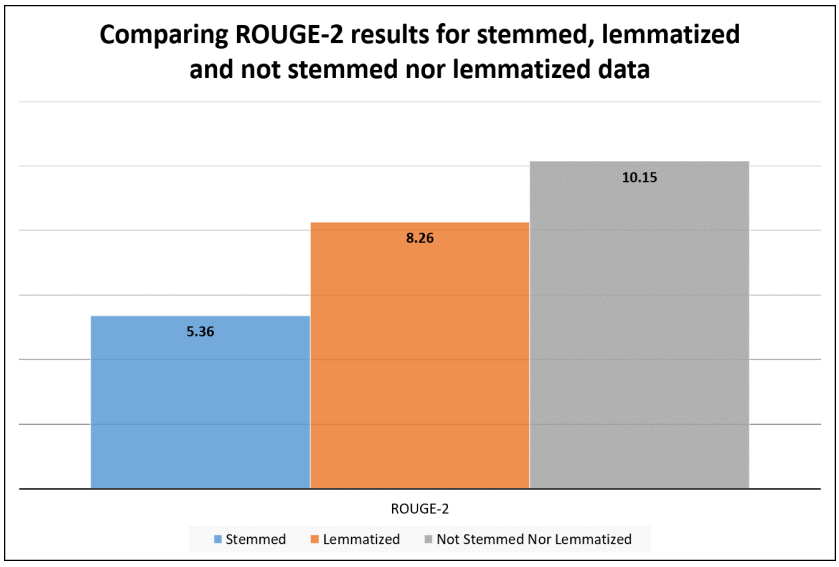

Figure 3: Comparison between training the AraBART model on stemmed data, lemmatized data and neither stemmed nor lemmatized data. The figure depicts the average Rouge-2 on the test dataset. This shows how stemming and lemmatization have negative impact on the properness of the produced summaries.

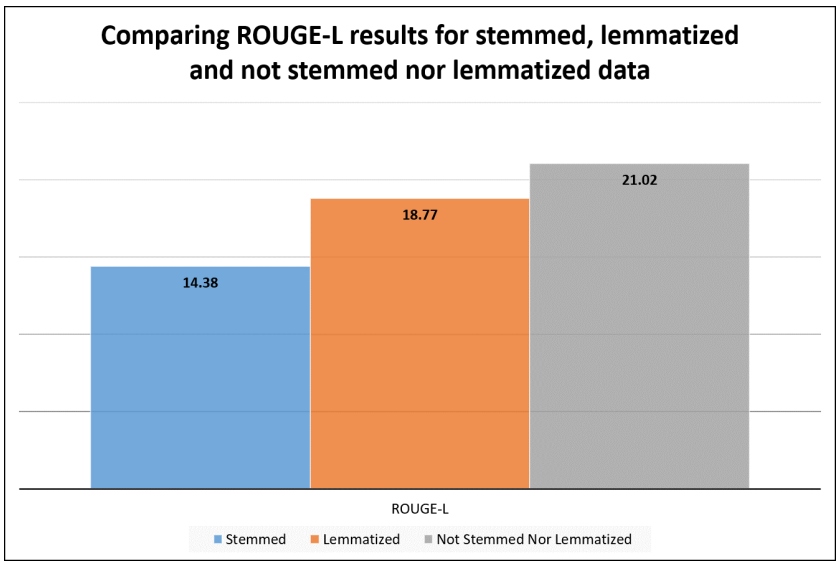

Figure 4: Comparison between training the AraBART model on stemmed data, lemmatized data and neither stemmed nor lemmatized data. The figure depicts the average Rouge-L on the test dataset. This shows how stemming and lemmatization have negative impact on the properness of the produced summaries.

lems similar to those caused by lemmatization. In addition to that, a word like استعمر "colonised," for instance, is reduced to its base عمر "age," which has a completely different meaning than the original word. This change has the potential to confound the model and produce summaries that differ greatly from the intended meanings. Therefore, lemmatization (training setup 2) and stemming (training setup 1) both produce worse results than the method without stemming or lemmatization (training setup 3).

5 Conclusion

In this study, three different setups of preprocessed data were used to fine-tune and train the AraBART model for Arabic automatic text summarization. The three different setups were: training setup #1 preprocessing involved stemming, training setup #2 preprocessing involved lemmatization, and training setup #3 preprocessing neither involved stemming nor lemmatization. In order to assess the effects of stemming and lemmatization on the summarizing task, it was necessary to examine how well each configuration performed on the testing data segment. In this work, the assessment metrics ROUGE-1, ROUGE-2, and ROUGE-L were applied. Training configuration #3, which did not use stemming or lemmatization, produced the best outcomes in terms of all metrics. This suggests that Arabic automatic text summarization utilizing the AraBART model was negatively impacted by stemming and lemmatization. The alteration of word structures can be blamed for the negative outcomes seen in lemmatization and stemming. Lemmatization eliminates affixes and may cause the loss of crucial information, such as plurality, which may confuse the model and result in the production of summaries devoid of crucial information. Stemming can change words into entirely different meanings and remove plurality, which makes it more difficult for the model to produce correct summaries.

References

[AFA12] Fahad Alotaiby, Salah Foda, and Ibrahim Alkharashi. New approaches to automatic headline generation for arabic documents. *Journal of Engineering and Computer Innovations*, 3(1):11–25, 2012.

[AGN+22] Anas Alokla, Walaa Gad, Waleed Nazih, Mustafa Aref, and Abdel-badeeh Salem. Pseudocode generation from source code using the bart model. *Mathematics*, 10(21):3967, 2022.

[ATAO14] Ahmad T Al-Taani and Maha M Al-Omour. An extractive graph-based arabic text summarization approach. In *The International Arab Conference on Information Technology*, pages 158–163, 2014.

[BGE+20] Asmaa A Bialy, Marwa A Gaheen, RM ElEraky, AF ElGamal, and Ahmed A Ewees. Single arabic document summarization using natural language processing technique. *Recent Advances in NLP: The Case of Arabic Language*, pages 17–37, 2020.

[BLY14] Vimala Balakrishnan and Ethel Lloyd-Yemoh. Stemming and lemmatization: A comparison of retrieval performances. 2014.

[BQWR14] Yanwei Bao, Changqin Quan, Lijuan Wang, and Fuji Ren. The role of pre-processing in twitter sentiment analysis. In *Intelligent Computing Methodologies: 10th International Conference, ICIC 2014, Taiyuan, China, August 3-6, 2014. Proceedings 10*, pages 615–624. Springer, 2014.

[EBEK20] Reda Elbarougy, Gamal Behery, and Akram El Khatib. A proposed natural language processing preprocessing procedures for enhancing arabic text summarization. *Recent Advances in NLP: The Case of Arabic Language*, pages 39–57, 2020.

[ETH+22] Moussa Kamal Eddine, Nadi Tomeh, Nizar Habash, Joseph Le Roux, and Michalis Vazirgiannis. Arabart: a pretrained arabic sequence-to-sequence model for abstractive summarization. *arXiv preprint arXiv:2203.10945*, 2022.

[GSK18] Partha Protim Ghosh, Rezvi Shahariar, and Muhammad Asif Hossain Khan. A rule based extractive text summarization technique for bangla news documents. *International Journal of Modern Education and Computer Science*, 10(12):44, 2018.

[HBI+21] Tahmid Hasan, Abhik Bhattacharjee, Md Saiful Islam, Kazi Samin, Yuan-Fang Li, Yong-Bin Kang, M Sohel Rahman, and Rifat Shahriyar. Xl-sum: Large-scale multilingual abstractive summarization for 44 languages. *arXiv preprint arXiv:2106.13822*, 2021.

[JBB21] Deepali Jain, Malaya Dutta Borah, and Anupam Biswas. Summarization of legal documents: Where are we now and the way forward. *Computer Science Review*, 40:100388, 2021.

[KM00] Kevin Knight and Daniel Marcu. Statistics-based summarization-step one: Sentence compression. *AAAI/IAAI*, 2000:703–710, 2000.

[KSND21] Divya Khyani, BS Siddhartha, NM Niveditha, and BM Divya. An interpretation of lemmatization and stemming in natural language processing. *Journal of University of Shanghai for Science and Technology*, 22(10):350–357, 2021.

[LLG+19] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[PBS23] Krutika Patil, Medha Badamikar, and Sheetal Sonawane. Nlp based text summarization of fintech rfps. In *2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, pages 865–869. IEEE, 2023.

[SBS14] Md Arafat Sultan, Steven Bethard, and Tamara Sumner. Towards automatic identification of core concepts in educational resources. In *IEEE/ACM Joint Conference on Digital Libraries*, pages 379–388. IEEE, 2014.

[TM12] Juan-Manuel Torres-Moreno. Beyond stemming and lemmatization: Ultra-stemming to improve automatic text summarization. *arXiv preprint arXiv:1209.3126*, 2012.

[TTJ06] Michal Toman, Roman Tesar, and Karel Jezek. Influence of word normalization on text classification. *Proceedings of InSciT*, 4:354–358, 2006.