Arabic QA DistilBERT Fine-Tuning

Mohamed Abdelgaber, Ali Helmy, Mahmoud Nabil

May 7, 2023

1 Introduction

Natural Language Processing (NLP) is a rapidly growing field of artificial intelligence that focuses on enabling computers to understand and analyze human language. One important task in NLP is Question Answering (QA), which involves answering natural language questions posed by users based on a given context. In this project, we aim to build an AI model that can perform QA in Arabic language using the fine-tuned DistilBert model.

Arabic is a complex language with its own unique characteristics, which makes it challenging to build accurate NLP models for it[GSA+21]. Nevertheless, there is a growing demand for Arabic NLP applications in various fields, such as education, healthcare, and finance. Thus, our project can contribute to the development of Arabic NLP by providing a QA model that can accurately and efficiently answer questions in Arabic language.

To achieve our goal, we will use the Arabic Questions and Answers Dataset (AQAD)[AMS+20] after preprocessing it. We make use of the Arabert[ABH] preprocessor and tokenizer in the preprocessing stage to tokenize the text and prepare it for fine-tuning the DistilBert model.

The rest of this report will describe the data analysis we conducted, the data collection and preprocessing steps, and the system architecture of our QA model. Additionally, we will discuss the challenges we faced during the project and our initial plans for addressing them.

2 Data Analysis

The dataset used in this project is the Arabic Questions and Answers Dataset (AQAD) that contains the questions, answers, and the context from which the answer is pulled. The data consitis of 17911 questions and follows the structure of the Stanford Question Answering Dataset (SQuAD)[RZLL16].

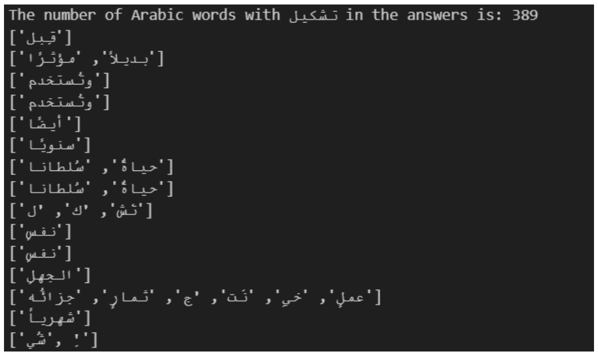
2.1 Data Frame

The dataset is put into a dataframe with 4 columns and N rows:

1. title,question,answer and is-impossible columns.
2. Each row represents an entry (QA).

2.2 Analysis

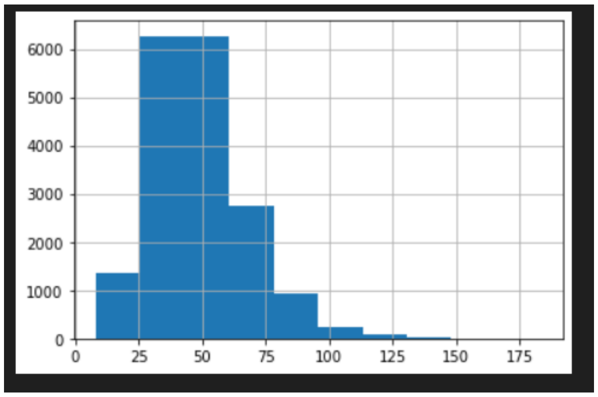
1. We started analysing the dataframe by getting the count of the topics in the table ”title” , counting the words that have diacritics in questions and answers, counting the English words, and presenting some of these words. We Found that words that have diacritics represent small numbers and that there are few english words . We used the popular english words to detect the english words present in the dataset so some non-frequently used english words may have not been detected.



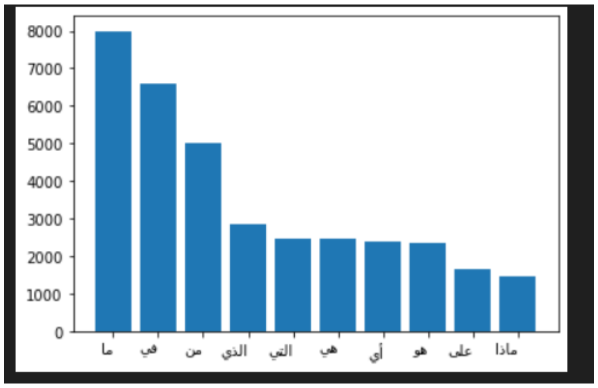
```
The number of Arabic words with تشكيل in the answers is: 389
['قِبل']
['بـديـلاً', 'مـؤثـرًا']
['وتُستخدم']
['وتُستخدم']
['أيضًا']
['سنويًا']
['حياةً', 'سُلطانا']
['حياةً', 'سُلطانا']
['تَشّ', 'ك', 'ل']
['نفسِ']
['نفسِ']
['الجهلِ']
['عملٍ', 'خيٍ', 'نَت', 'ج', 'ثمارٍ', 'جزائُه']
['شهرياً']
['شيٍّ', '!']
```

2. To proceed with the analysis, the answers that have no values are removed, and we calculated the average question and answer length. We found that answers are more likely to be shorter than questions.

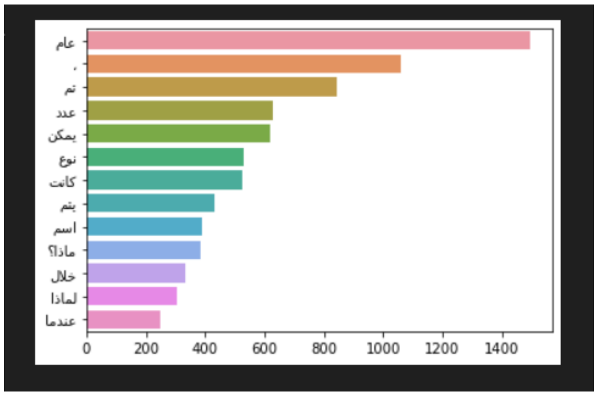
3. Then we showed the number of characters, words, and average word length present in each question and answer using histograms.



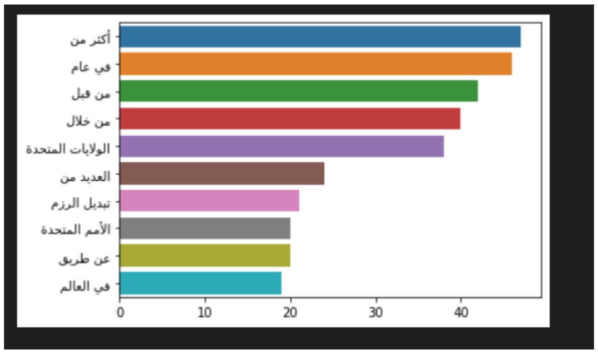
4. Then we used a function to get Arabic stopwords to check their frequencies in the questions and answers, and as expected, the frequencies are very high and words like ”في” and ”ما” has the highest frequencies.



5. Then we checked what words other than the stop words are the most frequent in questions and answers. ”عام” is the most frequent word which indicate that most asked questions in dataset about the year.



6. Finally, we used a function to get the top n-grams in QAs, and we checked for bi-grams and tri-grams in QAs. Most Frequent Bigram in questions are ”ما هو” and ”ما هي”.



3 Data Preprocessing

In this section, we will discuss the data preprocessing steps that we have taken to prepare our dataset for training our Question Answering model.

After reading and parsing the dataset, it is split into train-test splits with 80:20 ratio. The conducted analysis showed that our dataset has a good variety of questions, contexts and answers to ensure that our model is trained to handle a variety of inputs.

To prepare our dataset for training, we have to preprocess it. The following are the steps that we followed in our preprocessing stage.

3.1 Adding Answer End Index

We first processed the answers list of dictionaries to add the key 'answer_end' key with its value to every answer in our dataset. The 'answer_end' indicates the index of the final character of the answer inside the context for a specific question. This step is important because we will use this index to identify the end of the answer during training. The 'answer_end' value is calculated using the 'answer_start' value in addition to the answer length. We'd then compare the answer given in the original dataset with the answer produced from the substring of the context with the 'answer_start' and 'answer_end' indices. In case the answers don't match, the 'answer_end' value is adjusted to meet this criteria.

3.2 Removing Diacritics and Tatweel

Initial experiments with the Arabert tokenizer showed that removing the diacritics and Tatweel reduced the number of '[UNK]' tokens produced from the tokenizer. This is a recommended procedure for Arabert since the same processing is done on the data used to train the tokenizer.

3.3 Using Farasa segmentation

we used Farasa[ADDM16] to segment the Arabic text as recommended for the Arabert tokenizer. Farasa, is a fast and accurate Arabic segmenter that facilitates capturing the details of the morphologically rich Arabic language. Farasa segments the words based on a large number of features and is used in the Arabert tokenizer training.

3.4 Tokenization

We used the pre-trained Arabert Arabic tokenizer to tokenize the Arabic text. This tokenizer is of word-piece type. Word-piece tokenizers are a type of subword tokenization technique commonly used in natural language processing (NLP) tasks. They aim to break down words into smaller subword units called ”word-pieces” based on linguistic patterns and frequency statistics observed in a given corpus. The word-piece arabert tokenizer often captured more characters per token proving more efficient compared to character-based tokenizers. This is especially useful when feeding data to the Bert-based models like Distilbert that has a small embedding size of 512 tokens. The ’Padding’ and ’Truncation’ flags are set to ’True’ to make sure the input tokens match the embedding size of the tokenizer.

3.5 Concatenation

After tokenization, we concatenated each context and question to form a single input sequence. This is because our model expects each context and question pair together to produce the answer.

3.6 Adding Answer Start and End Indices to the tokenization

After that, we added the answer start and end indices to the encoding after converting from character to token. This step is crucial because it allows our model to learn where the answer is located within the context without concatenating the answer to the context and question.

4 System Architecture

The system architecture for the project is based on the fine-tuning of the DistilBert model for the task of Question Answering in the Arabic language. DistilBert is a smaller and faster version of BERT, which makes it easier to train and more efficient to use for QA tasks.

The input to the model is a concatenated string of the context and question, which is then tokenized using the Arabert tokenizer. The tokenization process converts the input text into a series of tokens, which are then mapped to their corresponding indices in the model's embedding layer.

After tokenization, the answer start and end indices are added to the tokenized encoding. This is done to inform the model about the boundaries of the answer in the context.

The model architecture consists of multiple layers, including embedding, attention, and feed-forward layers. The embedding layer converts the tokenized input into a dense vector representation. The attention layer helps the model to focus on the relevant parts of the input during training and inference. The feed-forward layer maps the output of the attention layer to the final predicted answer.

During training, the model is optimized using the Adam optimizer, which adjusts the learning rate based on the gradient of the loss function. The loss function used in this project is the mean squared error (MSE), which measures the difference between the predicted answer and the true answer.

Once the model is trained, it is used for inference by feeding it with a new context and question. The model outputs the predicted answer, which can be compared with the true answer to evaluate the performance of the model.

Overall, the system architecture for this project is designed to enable the fine-tuning of the DistilBert model for the task of Question Answering in Arabic language. The architecture includes several key components, including tokenization, encoding, attention, and optimization, which work together to enable the model to learn how to answer questions based on the given context.

5 Conclusion

In conclusion, this project aimed to produce an AI model that can perform the task of Question Answering in the Arabic language. The model is trained on a dataset that contained questions, their answers, and the context from which the answers are derived. The dataset is split into training and testing sets. To preprocess the data, diacritics and tatweel are removed from the Arabic sentences, the text is segmented with farasa, and a pre-trained tokenizer called Arabert tokenizer is used to tokenize the text. The tokenizer used is of the word-piece type. After tokenizing, the contexts and questions are concatenated, and the answer start and end indices are added to the encoding after converting from characters to tokens. The model is fine-tuned on DistilBert, a pre-trained transformer-based model for natural language processing.

References

[ABH] Wissam Antoun, Fady Baly, and Hazem Hajj. Arabert: Transformer-based model for arabic language understanding. In *LREC 2020 Workshop Language Resources and Evaluation Conference 11–16 May 2020*, page 9.

[ADDM16] Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California, June 2016. Association for Computational Linguistics.

[AMS+20] Atef Adel, Bassam Mattar, Sandra Sherif, Eman Elrefai, and Marwan Torki. Aqad: 17,000+ arabic questions for machine comprehension of text. 11 2020.

[GSA+21] Imane Guellil, Houda Saâdane, Faical Azouaou, Billel Gueni, and Damien Nouvel. Arabic natural language processing: An overview. *Journal of King Saud University - Computer and Information Sciences*, 33(5):497–507, jun 2021.

[RZLL16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.