Religious Text Analysis and ChatBot Transformers-Based Models: مَرجِعنا

Yousif Adel Khalil[1], Mostafa Samer Dorrah[2], Abdallah Ashraf Elsaadany[3],
Ahmed Ashraf Mohamed[4], Mahmoud Mohamed Mahmoud[5], Mariam Wael Mohamed[6]
ITCS-AI Nile University
Under the Supervision of:
[1]Dr. Ghada Khoriba
[2]Eng. Salma Gamal
AIS302

***Keywords*— GPT, araGPT, Arabic NLU, Islamic Fatwa, araBERT**

I. Introduction

While there are more than 313 million Arabic speakers in the world, and more than 1.9 billion Muslims, there is a lack of true guidance in regards of the Sharia of Islam and most importantly in the provisions of pillars of Islam such as the prayer, fasting, etc.

Nowadays, the internet is filled with misleading fatwas and the Muslim who really searching for the true information which is based on the Quran or on the Hadith of Prophet Mohamed is in a problem, he/she has to search about who gave the fatwa and to check the validity of the Hadith that the Sheikh relied on in his fatwa.

In 'مرجعنا', we will make a GPT2 bot that can give a reliable and valid fatwa. We will use transformers (GPT2) for the bot to give fatwa because we need it to give "human-like" responses. The model will be mainly and initially trained on islamweb.net fatwas, it is a famous and valid source where any fatwa is given by a knowledgeable and reliable Sheikh with the Quran and trusted Hadith. The bot is going to be initially in Arabic language only.

II. Background

A. Related Work

Our work is related to prior work in both Quranic research and Question Answering systems. a) Quranic Research: Several studies have been made to understand the Quranic text and extract knowledge from it using computational linguistics. Saad et al. (2009) proposed a simple methodology for automatic extraction of concepts based on the Quran in order to build an ontology. In (Saad et al., 2010), they developed a framework for automated generation of Islamic knowledge concrete concepts that exist in the holy Quran. Qurany (Abbas, 2009) builds a Quran corpus augmented with a conceptual ontology, taken from a recognized expert source 'Mushaf Al Tajweed'. Quranic Arabic Corpus (Atwell et al., 2011) also builds a Quranic ontology of concepts based on the knowledge contained in traditional sources of Quranic analysis, including the sayings of the prophet Muhammad (PBUH). Many studies have been conducted to retrieve answers from the Holy Quran for the user's questions using information retrieval techniques such as semantic similarity and pattern matching. Abdelnasser et al. (2014) developed Al-Bayan system to answer Arabic Quranic questions. Al-Bayan represents verses by concept vectors and then uses cosine semantic similarity to retrieve the related verses for the user query. The system achieved 65% in terms of accuracy. In addition, Alqahtani (2019) built a model to answer the Arabic Quranic questions. This model enriched the user query with semantic features using the word2Vec algorithm. Then it extracted the most related concepts to the query from Quranic ontology using the cosine similarity. After that, it displayed the verses of the matched concepts to the user. The evaluation results 121 showed 41% in terms of recall. These studies have contributed significantly to enriching the field of Quranic research. However, they can extract the required answer from the verse for only specific types of questions or answer the questions with the whole verse.

b) Question Answering (QA) Systems: the Arabic QA research is still limited in terms of accuracy. Some Arabic systems have been proposed such as: QARAB (Hammo et al., 2002) which is a QA system that takes factoid Arabic questions and attempts to provide short answers. ArabiQA (Benajiba et al., 2007) which is fully oriented to the modern Arabic language. It also answers factoid questions using Named Entity Recognition. However, this system is not completed yet. This system proposed a new approach which can answer questions with multiple answer choices from short Arabic texts. However, its overall accuracy is 0.19. Also, all these systems target the modern standard Arabic. To the best of our knowledge, no previous research was proposed for the Quranic classical Arabic question answering.

B. Algorithms, Techniques, & Tools

In our project we used some needed libraries from data cleaning and modeling such as,

```
from transformers import GPT2LMHeadModel, GPT2LMHeadModel
from arabert.aragpt2.grover.modeling_gpt2 import GPT2LMHeadModel
```

```
from torch.utils.data import Dataset, DataLoader
from transformers import AutoTokenizer
from transformers import TextDataset,DataCollatorForLanguageModeling
from transformers import Trainer, TrainingArguments,
AutoModelWithLMHead
```

In data cleaning: we dropped off the ID feature, the rows which contains un-accessible resource,which is not useful in our case,and Creating a function to remove any special or random charachter that is apart from the arabic charachters, numbers,
punctuation, and brackets in ['ans'] column, Dropping every row that contains an answer with length less than 118 character.
In Model Intialization & Data Adjustement for Model Fine-Tuning: we used (ARAGPT2) because it is trained on the same large Arabic Dataset as AraBERTv2. It trained from scratch on a large Arabic corpus of internet text and news articles. It has 1.46 billion parameters, which makes it the largest Arabic language model available. The Mega model was evaluated and showed success on different tasks including synthetic news generation, and zero-shot question answering. Finally,we decided to publish our idea for all users,so we used gradio for interface and connected with our model to make it easier and affordable.

III. Methodology

In this project, we used islamweb.net data that we scraped as we mentioned in the Introduction section, we cleaned the data from NULL values, irrelevant text, and text that contains a reference that cannot be accessed. Then, we concatenate the question/title with the answer in one single column. After all the data cleaning we end up with 40,859 data-points (pair of question and answer), for computational cost reasons, we only select the first 20,000 data-point, 90% for training and 10% for validation. the data is in csv format, but for gpt2 (text generation task) it is better to feed the transformer text file, so we convert the csv file to text file, one for training and one for validation. We then fine-tuned the pre-trained model araGPT2-base (by aubmindlab) to our data, using their tokenizer. Also, for computational cost reasons, we used the base version of the transformer. special token added for the tokenizer (<question:>, <answer:>) to let the model understand which part is question and which part is answer. We created a function 'load_dataset()' that loads the training and validation dataset, tokenizes it, and pad the input with 'DataCollatorForLanguageModeling()' function. The Majority of the functions and methods used are from the 'transformers' huggingface library. The number of trainable parameters are 134,994432 parameters. The most important training arguments used for the training part are as follows:

1- num_train_epoch: Set to 30 epochs

2- per_device_train_batch_size: set to 20 batches per epoch

3- per_device_eval_batch_size: set to 10 batches per epoch

4- save_steps: set to 20000

5- do_eval: set to True to enable validation during training

6- evaluation_strategy: set to 'epoch' to validate on the validation set after each epoch

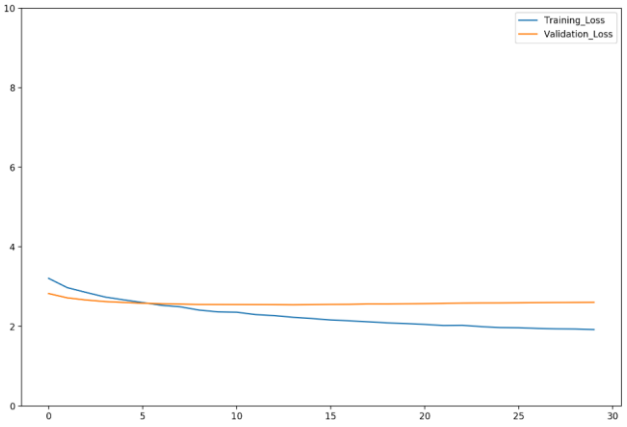
Then we give the trainer the model, the arguments, data_collator, train and validation dataset. we tracked Training loss and validation loss only during the training. The model is trained on a single GPU (NVIDIA GeForce RTX 3070 8GB GDDR6), the training/validation process took 8 hours and 20 minutes.

After the fine-tuning the model, the weights are saved in the local device to use the model afterwards without the need of training again. To deploy the model, 'gradio' library has been used, the user asks a question, the answer then appears on the right side of the web-page in a text box. the maximum length of a text that the model can generate is set to 300.

Last but not least, there is a hold-out-test-set that the model must be tested on. the hold-out-test-set contains 15 questions related to the Islamic field, the model have to generate responses for the 15 questions. Perplexity meausure is used to determine the performance of the model on a new unseen data, perplexity is a common metric used for language models and it measures the wellness of a model or a probability distribution on predicting the next word or phrase. In other words, when perplexity is high, it means that the level of confidence of the model while predicting the next words is low. And when the perplexity is low, the model's confidence while predicting the next word is high.

IV. Results & analysis

Before, we fine-tuned the model where no special tokens are provided, the model started with training loss of 4.9 and is learning the data slowly. Therefore, we tried to add special tokens to improve the model's performance. The model started with training loss of 3.2 when special tokens are provided and is learning much faster. The model finished the training/validation process with training loss of 1.9 and validation loss of 2.6, it is reasonably great due to computational and data limitations, in addition that there is no sign of over-fitting nor under-fitting as shown in the figure below.



Now it's time to test the fine-tuned model on unseen data and that is the hold-out-test-set we have prepared; it contains 15 questions that any random person might ask, and it is relatively new to the model. We used the perplexity measure as we discussed in the Methodology section of the documentation. The mean perplexity scores the fine-tuned model got on the 15 unseen questions is 2.3. The perplexity score is very low, and it is good news for us. However, we have noticed that the model got low perplexity scores on the majority of questions related to the Islamic Religion, and that is understandable because our fine-tuned model is closed domain and is very specific in terms of the knowledge the model has. So, almost any question related to Islam and Islamic Sharia and whatsoever asked to the fine-tuned model it is going to get a low perplexity score. Nevertheless, when the model generates a relatively perfect answer to the question asked the perplexity score is the lowest the score for the model, and if the answer has some manipulated text and does not answer the question directly it will get a higher perplexity score from the one before we talked about, and if the answer is completely does not answer the question and there is some verses of the Quran or Hadiths are manipulated the model would get the highest perplexity score of the 3 examples we stated. Thus, Perplexity score is a good measure to test a text-generation model like ours and we got good and informative results overall and after all.

V. Conclusion

In conclusion, we have successfully achieved our main aim that involve on providing true guidance in in the Sharia of Islam, specifically in relation to the pillars of Islam, including prayer and fasting. This significant achievement was made possible through the development of our generative model "فَتْوَى", which mostly is able to deliver accurate and reliable
fatwas. Our model was trained on authorized sources mainly on Islam web which is known for its credibility to ensure that the generated fatwas are authenticated. Our model will help Muslims from all over the world to answer their Islamic questions. but at the present time we cannot get used to it completely because it needs more training, and we must also bear in mind that it is not possible to completely rely on matters related to religion on an artificial intelligence model.

VI. Appendix

Web Scraping: Abdallah & Ahmed

Data Cleaning & Preprocessing: Yousif & Mostafa

GPT Model: Abdallah, Mostafa, & Yousif

Bert Model: Mariam & Mahmoud

Evaluation & Validation: Yousif

Information Retrieval Model & Text Similarity: Ahmed

Model Deployment: Mahmoud & Mariam

Model Interpretation: Ahmed

Team Members' IDs:

Yousif Adel Khalil (202000238)

Mostafa Samer Dorrah (202000125)

Abdallah Ashraf Elsaadany (202000467)

Ahmed Ashraf Mohamed (202000563)

Mahmoud Mohamed Mahmoud (202002556)

Mariam Wael Mohamed (202001259)

VII. REFERENCES

[1] https://huggingface.co/aubmindlab/aragpt2-base

[2] https://www.islamweb.net/ar/

[3] https://huggingface.co/docs/transformers/notebooks

[4] https://arxiv.org/pdf/2012.15520.pdf

[5] https://arxiv.org/pdf/2003.00104.pdf

[6] https://towardsdatascience.com/arabic-nlp-unique-challenges-and-their-solutions-d99e8a87893d

[7] https://huggingface.co/aubmindlab/bert-base-arabert

[8] https://www.ijcsmc.com/docs/papers/April2021/V10I4202119.pdf