Endsem Report

On

Conversational Multilingual Chatbots

Submitted by:-

Shashank Agarwal **2020A7PS0073P**

Abhijith Kannan **2019B5A70688P**

Submitted to:-

Dr. Yashvardhan Sharma

Associate Professor,

Computer Science and Information Systems Department

BITS-Pilani, Pilani Campus

Birla Institute of Technology and Science, Pilani (Rajasthan)- 333031

March, 2023

INDEX

BACKGROUND

We had designed a chatbot for question answering faq bot for monuments in the english language using the RASA framework. This performed well for small datasets with good accuracy, but wasn't scalable for larger datasets. RASA also didn't directly support the 'arabic' language.

Thus we had later used BERT and GPT-3 models which were pre-trained in large arabic datasets to implement the question answering models. The models we had implemented were "AraBert-Arabic-SQuADv2-QA", "Ara Electra-base-discriminator", "bert_qa_arap_qa_bert_large_v2', 'Bert_qa_arap_qa_bert_v2' and 'Bert_qa_arap_qa_bert'. These models were found to be scalable with reasonable accuracy. But the latest dataset provided by the company was relatively small and didn't have 'context' data available with it. The number of unique questions in the dataset was also relatively small.

Thus currently in our project we have decided to implement the question answering system using classification models (which are also pre-trained in the arabic language). And also use tree-based classification logic to categorize the questions in pre-defined genres.

MOTIVATION

This project is designed to be scalable in an enterprise environment. Thus this model is designed with all the constraints provided to us by the client (company).
The company wanted a faq chatbot application to cater to an ‘arabic’ speaking audience. They wanted the model to have good latency, having good scalability and also having a provision to edit the backend database dynamically. (insert/ delete/ update the question answer pairs).
Thus we have chosen a classification model using cross encoders. This was more practical for the scale of companies requirements and was also providing good accuracy.
The company required us to design two models to compare their performance.

1) Use an english based classification model. Translate the arabic input given by the user into english and feed it to the model. The model will use the english dataset provided by the company. The response will be translated back to arabic and presented to the user.
2) A pure arabic based classification model. The arabic input given by the user is directly fed into the model. Now the model works on the translated version of the dataset which was provided by the client. Paraphrasing of questions is taken care of by our model. The arabic response returned by the mode is directly returned to the user

The confidence of the result can also be shown to the client if required. Now if the confidence level of none of the answers is above a specified threshold (the query given by the user is not matching any of the questions in our predefined database), then we will revert to the “**fallback policy”** and request the client to seek expert help since the model can’t accurately answer that question. Now such an action can be programmed to trigger a notification to the manager to add the answer corresponding to the above unanswered question into the database for future reference. Thus we have implemented a sort of feedback policy into the model.

OBJECTIVE

To design a FAQ Question answering chatbot in the Arabic language which is scalable and works with the given dataset.
The chatbot enquires the user about the category of question the user wishes to ask, then it asks the user for the query. If the query is matching very similarly with multiple questions in the chatbot's database the bot will validate the user about the question and then give the correct answer to the user.

PROPOSED ALGORITHMS AND TECHNIQUES

The chatbot is a sophisticated system that requires access to a vast amount of data to provide appropriate responses. Typically, such data is provided in the form of a context-question-answer (SQUAD) dataset. However, the dataset provided lacked the context and only had question-answering pairs. Such data cannot be used for training natural language processing (NLP) models effectively. The absence of this data renders training any NLP model on such a dataset challenging.

One approach to overcome this limitation is to use a similarity-based approach that compares user queries to all relevant questions in the dataset to find the most similar one. This approach entails providing the same answer when the similarity between the user's query and the most similar question is above a set threshold.

To implement this similarity-based approach, we use both English and Arabic NLP models. For English, we employ the cross encoder from the **"stsb-distil roberta-base"** model, which is widely used in various NLP tasks, including sentence similarity measurement. For Arabic, we use the **“AraBert”** model, an open-source model explicitly designed for Arabic language processing.

To find similarities between a pair of strings in Arabic, we first find the tensor encodings of the strings using the AraBert model, we extract the embedding of the tensor and calculate the cosine similarity of these embeddings using the PyTorch cosine similarity function.

The use of English NLP models provides the advantage of access to better cross encoders that can accurately identify correlations. However, no such cross encoders are currently available for Arabic models. By utilizing both English and Arabic NLP models, we can provide effective chatbot responses in multiple languages.

In conclusion, the use of a similarity-based approach using both English and Arabic NLP models is an effective strategy for providing accurate chatbot responses. This approach is essential when training NLP models is not feasible due to the absence of contextual information in the dataset. Therefore, the approach described here provides a practical solution for designing chatbots that can be used in multiple languages.

DATASETS USED

Arabic Datasets

English dataset

The English dataset was provided by the company for us. By their instruction we have translated it to make our Arabic database for implementation of our ML model.

The following important columns available for us in the dataset

- Category of question

- Question
- Answer

ML MODELS USED

"AraBert" Model

The AraBert model is a state-of-the-art language model for Arabic language processing, developed by a team of researchers from the American University of Beirut and the Qatar Computing Research Institute. It is based on the BERT (Bidirectional Encoder Representations from Transformers) architecture, which has been shown to be effective for a wide range of NLP tasks. AraBert is pre-trained on a large corpus of Arabic text and can be fine-tuned on specific tasks such as sentiment analysis, named entity recognition, and question-answering.

Cross encoders are a type of encoder that are specifically designed for comparing pairs of sentences or text segments. Unlike traditional encoders, which produce a fixed-length vector representation of a single input, cross encoders take two inputs and produce a single vector representation that captures the relationship between the two inputs. Cross encoders are commonly used in tasks such as paraphrase detection, semantic similarity, and text classification.

"stsb-distil roberta-base" model

The "stsb-distil roberta-base" model mentioned is a pre-trained cross encoder that is specifically designed for measuring the semantic similarity between pairs of sentences. It is based on the distilRoBERTa model, a smaller and more efficient version of the RoBERTa model, which is itself an extension of the BERT model.

By combining the AraBert and cross encoder models, we can effectively compare a user's query to all relevant questions in the dataset and find the most similar one. This similarity-based approach allows us to provide accurate responses even when contextual information is not available in the dataset. Additionally, by utilizing both English and Arabic NLP models, we can provide effective chatbot responses in multiple languages.

EXPERIMENTATION

- **Chatbot Implementation using English as intermediate language**

  Using Cross Encoders from Roberta Base on Sample Queries.

```
from sentence_transformers.cross_encoder import CrossEncoder
model = CrossEncoder('cross-encoder/stsb-distilroberta-base')

scores = model.predict([["What methods scientists use to study black holes", "What is the event horizon of a black hole"],
["What methods scientists use to study black holes", "What is the difference between a stellar black hole and a supermassive black hole"],
["What methods scientists use to study black holes", "Can anything escape a black hole"],
["What methods scientists use to study black holes", "How do scientists research about black holes"]])
print(scores)
```

  Output :

```
[0.32458493 0.51508856 0.24248636 0.7792017 ]
```

  The variation of score value for a sample data is shown in the above demonstration. The similarity between the user query and the questions in our database is compared and the most similar response is returned to the client

(To facilitate this english intermediate language chatbot, the dataset used was in english language. The query of the user is translated from arabic to english and fed to the model. The response of the machine is translated back into arabic and returned to the client)

- **<u>Chatbot Implementation using Arabic as intermediate language</u>**

Implementation of this chatbot is done using the Arabert model.

```
from transformers import AutoTokenizer, AutoModel
from arabert.preprocess import ArabertPreprocessor
# from arabert.preprocess_arabert import never_split_tokens
from farasa.segmenter import FarasaSegmenter
import torch

arabert_tokenizer = AutoTokenizer.from_pretrained(
    "aubmindlab/bert-base-arabert",
    do_lower_case=False,
    do_basic_tokenize=True,
    )
arabert_model = AutoModel.from_pretrained("aubmindlab/bert-base-arabert")

farasa_segmenter = FarasaSegmenter(interactive=True)
```

The function to calculate the similarity value between two arabic strings

```
def calculateSimilarity(str1, str2):
    emb1 = findEmbedding(str1)
    emb2 = findEmbedding(str2)
    diff = (emb1.shape[0] - emb2.shape[0])
    #print(diff)
    if (diff > 0):
      emb2 = F.pad(input=emb2, pad=(0, 0, diff, 0), mode='constant', value=0)
    else:
      emb1 = F.pad(input=emb1, pad=(0, 0, -diff, 0), mode='constant', value=0)
    # print(emb1.shape)
    # print(emb2.shape)
    cos = torch.nn.CosineSimilarity(dim=0)
    cos_sim = cos(emb1, emb2)
    overall_similarity = torch.mean(cos_sim)
    #print(overall_similarity)
    return overall_similarity
```

This function is optimally iterated with the database to produce the most similar response to the client's question

The main function which compares the client's query to the filtered database and returns the optimal response along with the confidence value of the response.
If the user's query does not match with any of the questions in the database with the atleast the threshold value of confidence, then the model reverts to the **fallback policy**. The model answers that "sorry, I don't have an answer for you"

```
QUESTIONS = df.columns.values.tolist()[0]
ANSWER = df.columns.values.tolist()[1]
def targetFunction(query):
  max = 0.0
  response = ""
  for ind in df.index:
    # print(ind)
    # print(df[QUESTIONS][ind])
    question=df[QUESTIONS][ind]
    answer=df[ANSWER][ind]
    # print(answer)
    res = calculateSimilarity(query, question)
    if(res > max):
      response = answer
      max = res

  return {"RESPONSE": response,"CONFIDENCE" : max}


def fresh_function():
  val=""
  while(1):
    val=input("enter your query (in arabic)   : ")
    if(val=="close"):
      break;
    response1=targetFunction(val)
    if(response1["CONFIDENCE"]<0.3):
        print("sorry, I don't have an answer for you")
    else:
      print("RESPONSE" , response1["RESPONSE"])
      print("confidence value",response1["CONFIDENCE"])
  return
fresh_function()
```

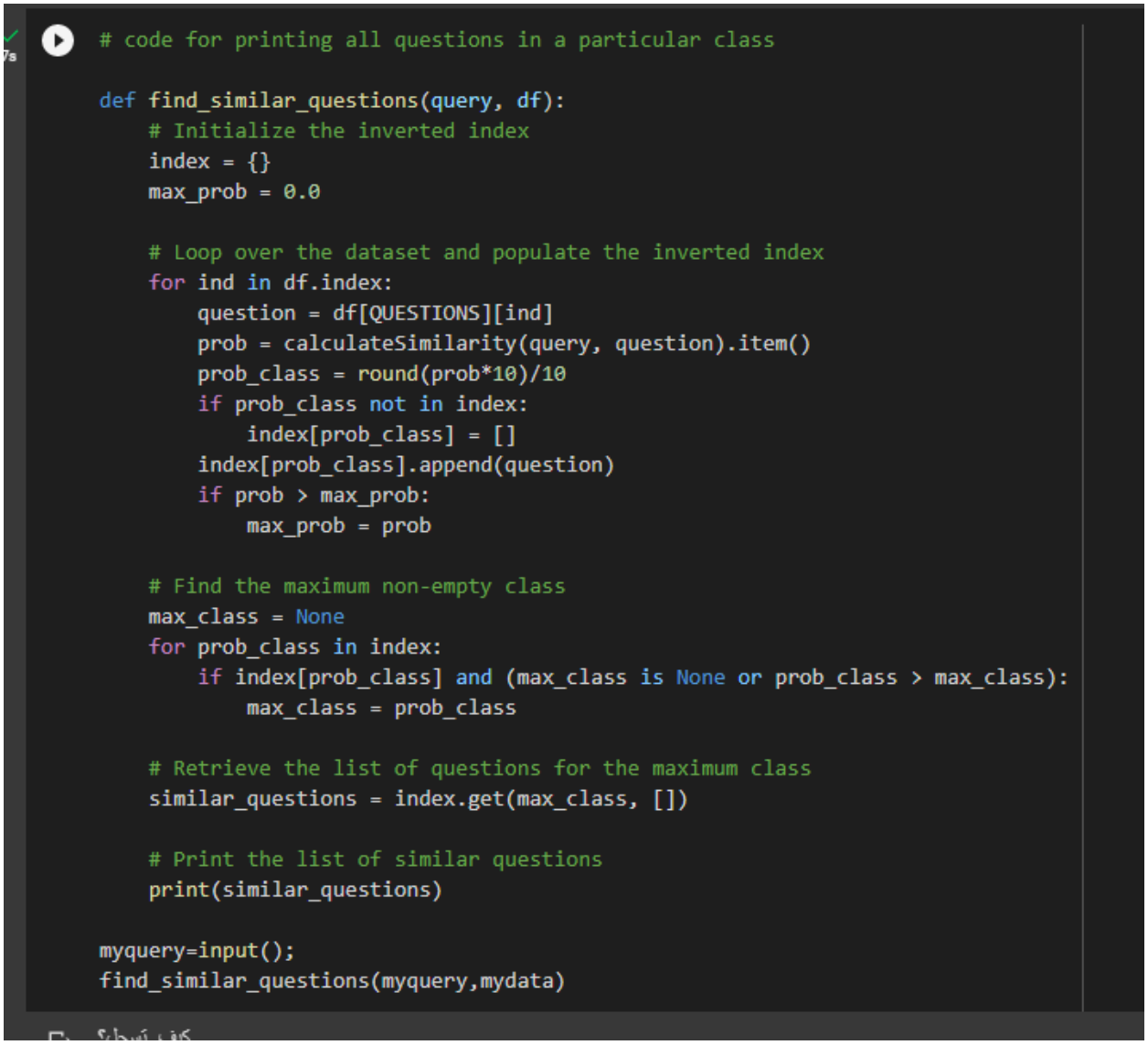
TREE-LOGIC IMPLEMENTATION

The linear implementation of the classification model was too slow and time-consuming. Since the model is comparing the query with all possible questions in our dataset.

```
#removed the third column
df1=df_without2.drop_duplicates(subset=['TAB3'],keep='first',inplace= False)
#print(df1['TAB1'])
df2=df1['TAB3']
df2=df2.dropna()
arr=df2.to_numpy()
#print(arr)
print("kindly pick one of the following options")
counter=1
response=-1
#print(type(str(counter)))
for i in arr:
  print("press " + str(counter) + " for " + i)
  counter=counter+1
response=input()
response= int(response)
#print(type(response))
print("your response is "+ str(response))
print("your response values is " + arr[response-1])
df_without3= df_without2.loc[mydata['TAB3']==arr[response-1]]
print(df_without3)
path="drive/MyDrive/Colab_Notebooks/downloads/filtered_dataset.xlsx"
df_without3.to_excel(path, index=False)
```

The above mentioned code dynamically read all the distinct categories present in the current database and enquires to the user which category of question the user wants to ask. Now the queries corresponding to the response of the user is stored in a different dataframe. Similary we iterate for all the 3 categories present in our dataset. Thus the filtered version of the dataset is created. Now this filtered dataset is pipelined to our model to query the classification model against.

Probability Classes Implementation

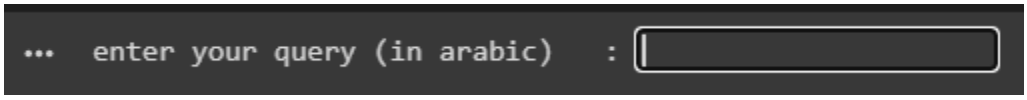


```
# code for printing all questions in a particular class

def find_similar_questions(query, df):
    # Initialize the inverted index
    index = {}
    max_prob = 0.0

    # Loop over the dataset and populate the inverted index
    for ind in df.index:
        question = df[QUESTIONS][ind]
        prob = calculateSimilarity(query, question).item()
        prob_class = round(prob*10)/10
        if prob_class not in index:
            index[prob_class] = []
        index[prob_class].append(question)
        if prob > max_prob:
            max_prob = prob

    # Find the maximum non-empty class
    max_class = None
    for prob_class in index:
        if index[prob_class] and (max_class is None or prob_class > max_class):
            max_class = prob_class

    # Retrieve the list of questions for the maximum class
    similar_questions = index.get(max_class, [])

    # Print the list of similar questions
    print(similar_questions)

myquery=input();
find_similar_questions(myquery,mydata)
```

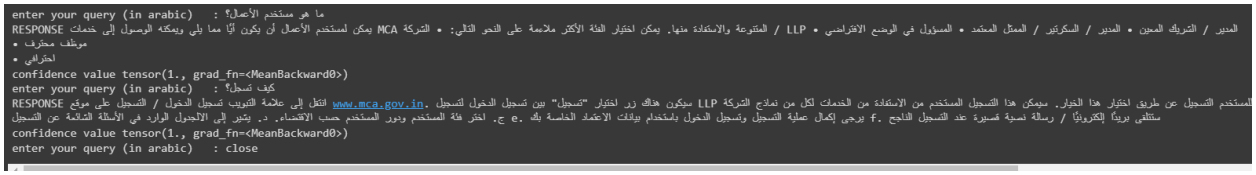
The above mentioned function implements the algorithm that if the confidence value of the best question related to the user's query and the second-best question is within the specified threshold. Then the model confirms with the user whether the user meant to ask this question or not. Thus we have implemented that if the probability of multiple questions belong in a class, then we ensure that the wrong answer is not displayed.

DEMONSTRATION

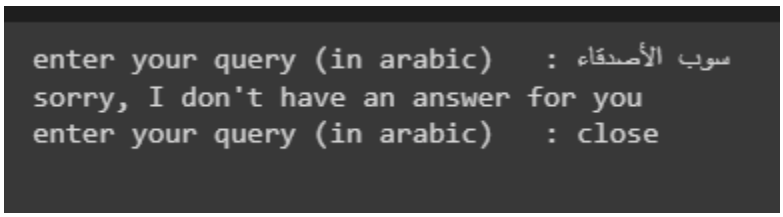
Interface for user to interact with the chatbot



Demonstration of Queries with user



Fallback Policy



UI DEMONSTRATION

QUERY – ذا نسيت كلمة المرور الخاصة بي ، فماذا أفعل؟

(What can I do If I forgot my passwords?)

Response: وانتقل إلى صفحة تسجيل الدخول. انقر فوق "هل نسيت MCA يرجى النقر فوق تسجيل الدخول / التسجيل الموجود على الصفحة الرئيسية لموقع ، LLP V3 لاستعادة معرف مستخدم PAN يرجى إنشاء شكوى متعلقة بالخدمة مع تفاصيل ، V2 معرف المستخدم؟" ربط الحاضرفوقحقل كلمة المرور واتبع الخطوات لاستعادة معرف المستخدم الخاص بك. لاستعادة معرف مستخدم وفقًا لذلك v2 وما إلى ذلك ، ثم يمكننا مشاركة معرف مستخدم DOB ومعرف البريد الإلكتروني و

Response in Arabic.

CONCLUSION

The use of a similarity-based approach using both English and Arabic NLP models is an effective strategy for providing accurate chatbot responses. This approach is essential when training NLP models is not feasible due to the absence of contextual information in the dataset. Therefore, the approach described here provides a practical solution for designing chatbots that can be used in multiple languages.

WORK UPDATES

WORK COMPLETED

- Link to BERT QA Implementation :
  Implementation of the English FAQ Bot. It uses cross encoders for finding the most similar query and gives the corresponding answer.

- Link to AraBERT QA Implementation :
  Implementation of the Arabic FAQ Bot. Created vector encodings of the query and used these encodings to find the similarity.

WORK REMAINING

- Extending the model for other languages and testing the model on them.

REFERENCES

- https://huggingface.co/cross-encoder/stsb-distilroberta-base
  Stsb-distil robert model for cross encoders.

- https://www.sbert.net/docs/pretrained_cross-encoders.html

- https://towardsdatascience.com/everything-you-need-to-know-about-albert-roberta-and-distilbert-11a74334b2da

- https://github.com/aub-mind/arabert
  Arabert Github repository

- https://huggingface.co/aubmindlab/bert-base-arabertv2