Natural Language Processing

Empathy-Driven Arabic Chatbot

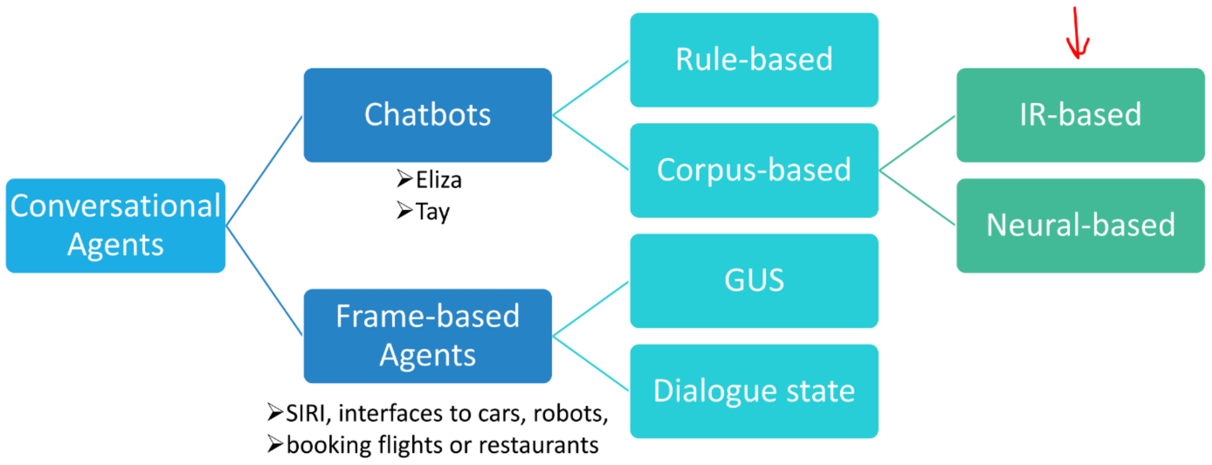
Submitted to:

Caroline Sabty

Submitted by:

Noor Alkendi 31-9103

Ghada Mansour 40-9240



Pipeline for Chatbot Building

1. Data Cleaning and Preparing
2. Preprocessing
   a. Ignore Some Words
   b. Tokenization
   c. Stemming
   d. Bag-of-Words
3. Supervised Sentiment Analysis using Multilayer Perceptron (MLP) Neural Networks Model for Deep Learning
4. Utterance Prediction based on Term Frequency — Inverse Document Frequency (**tf-idf**) and Cosine Similarity
5. Testing the Chatbot

So, the Chatbot is Information Retrieval Base

The Applied Pipeline for Chatbot Building

1. Data Cleaning and Preparing

The Dataset contains only two Contexts which are "**Afraid**" and "**Proud**" together with their different Prompts (the one in English and the Translated one), Utterances (the one in English and the Translated one), conv_id, utterance_idx, speaker_idx and other columns that showed no values like selfeval and others.

a. First, we drop all the unwanted Columns. So, we ended up having the utterance_idx, context, prompt (the one in Arabic), and utterance (the one in Arabic).
b. After that we check for the presence of null values to clean the data

2. Preprocessing

Then some preprocessing is applied on the Data to prepare it for Bag-of-Words and Modeling. Here we have:

a. Ignore some words: the ignore_words function in this step ensures removing the English words, Numbers, and Punctuations. Removing Punctuations is done before Tokenization since Punctuations in Arabic are not part of the words, unlike English which has apostrophes that may represent part of the word meaning (n't, s')
b. Tokenization: for splitting the words into an array
c. Stemming: for this step NLTK ISRIStemmer was used. We found many Arabic Stemmers; however, we chose this one based on Almazrua, et al., (2020), ISRIStemmer is one of the Top 2 Arabic Stemmers

   The preprocessing function takes a sentence and applies all of these steps to prepare it for the Bag_of_words

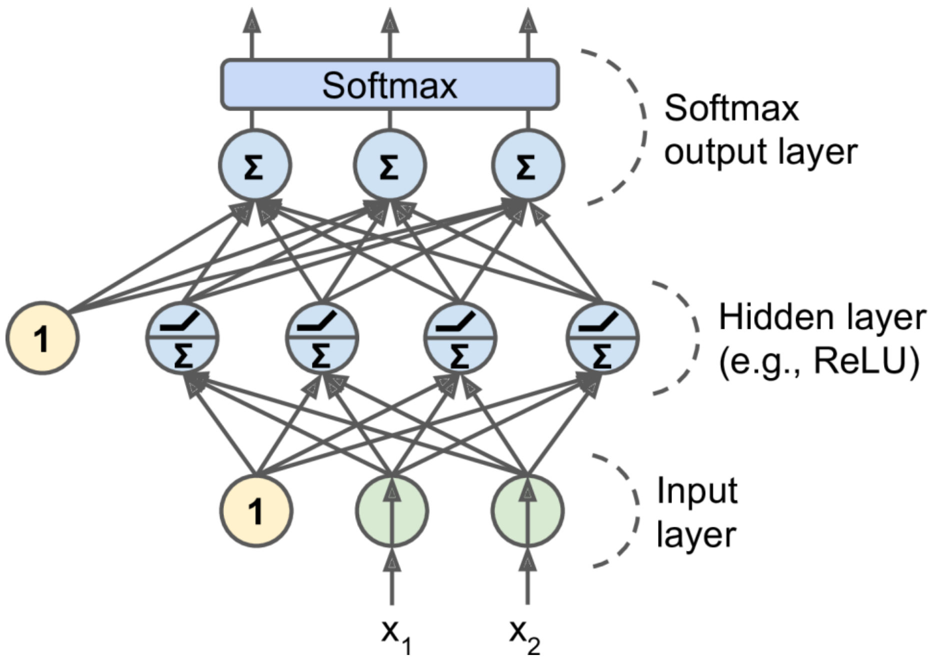
d. Bag_of_Words: to return a 0 and 1 vector with lengths equals to the all_words array representing the occurrences of a Sentence words in the all_words array.

Removing Arabic Stop words is not a part of Preprocessing since it resulted in lower Utterance Prediction Accuracy after testing it.

3. Supervised Sentiment Analysis using Multilayer Perceptron (MLP) Neural Networks Model for Deep Learning

By applying Multilayer Perceptron (MLP) Neural Networks Model, the Context was successfully predicted.

Here the Model takes the Bag_of_Words of each (input text) Utterance in the Dataset as the X part. So, all of them are appended to each other in the x_train. It also takes an array of all the distinct Contexts as the different number of classes (Labels) available which is 2 here (y_train). Here the y-train actually contains theses Contexts as indices (numbers)

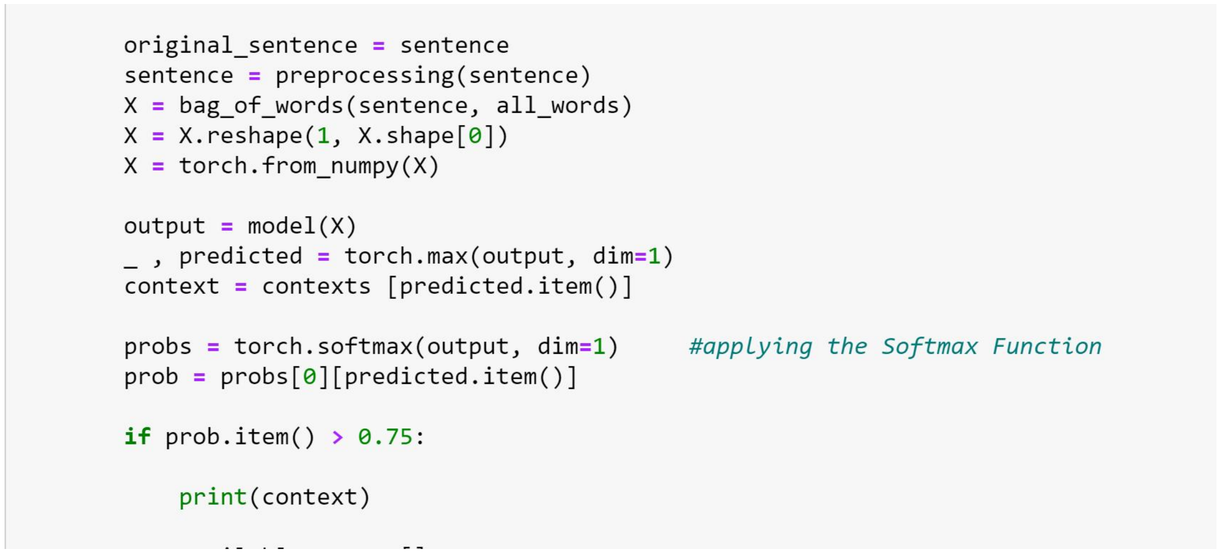


https://learning.oreilly.com/library/view/neural-networks-and/9781492037354/ch01.html

After creating a Dataset Loader and Training the Model on the Dataset by iterating over 1000 epochs using the mentioned Hyperparameters, a loss value of 0.000 was returned.

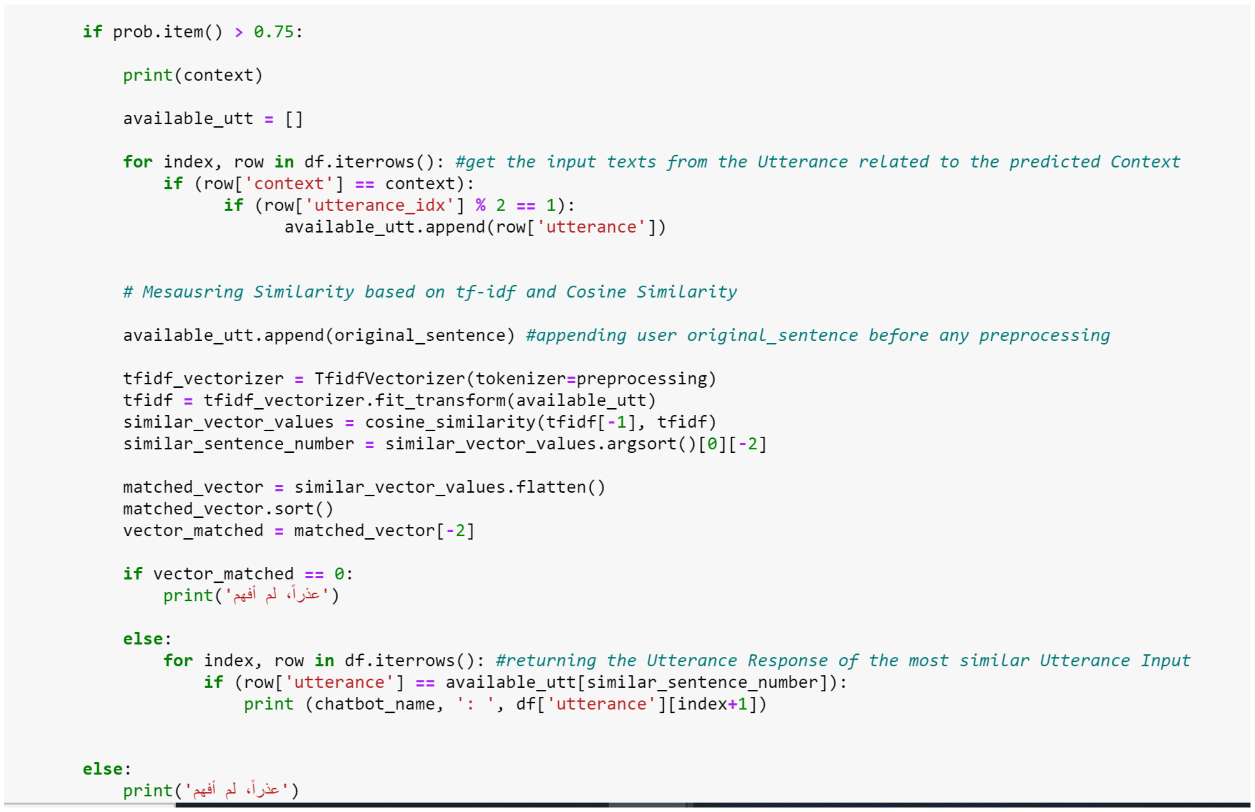
Context (Sentiment) Prediction

Then, we save the Model Data to be used further in prediction. The below part uses the Model with its data to output the Context Prediction and it applies the SoftMax Function on the Output. Actually, the Model outputs different real-valued scores for each Class (Label). So, in order to be able to make use of these Scores, SoftMax Function is used since it turns these scores into Probability Values. If the value (probability) was > 0.7 (meaning high portability). The Algorithm starts to measure the Similarity between the User Input Text and all of the (input text) Utterances related to this predicted Context.



```
original_sentence = sentence
sentence = preprocessing(sentence)
X = bag_of_words(sentence, all_words)
X = X.reshape(1, X.shape[0])
X = torch.from_numpy(X)

output = model(X)
_ , predicted = torch.max(output, dim=1)
context = contexts [predicted.item()]

probs = torch.softmax(output, dim=1)     #applying the Softmax Function
prob = probs[0][predicted.item()]

if prob.item() > 0.75:

    print(context)
```

4. Utterance Prediction based on Term Frequency — Inverse Document Frequency (tf-idf) and Cosine Similarity



```python
if prob.item() > 0.75:

    print(context)

    available_utt = []

    for index, row in df.iterrows(): #get the input texts from the Utterance related to the predicted Context
        if (row['context'] == context):
              if (row['utterance_idx'] % 2 == 1):
                    available_utt.append(row['utterance'])


    # Mesausring Similarity based on tf-idf and Cosine Similarity

    available_utt.append(original_sentence) #appending user original_sentence before any preprocessing

    tfidf_vectorizer = TfidfVectorizer(tokenizer=preprocessing)
    tfidf = tfidf_vectorizer.fit_transform(available_utt)
    similar_vector_values = cosine_similarity(tfidf[-1], tfidf)
    similar_sentence_number = similar_vector_values.argsort()[0][-2]

    matched_vector = similar_vector_values.flatten()
    matched_vector.sort()
    vector_matched = matched_vector[-2]

    if vector_matched == 0:
        print('عذراً، لم أفهم')

    else:
        for index, row in df.iterrows(): #returning the Utterance Response of the most similar Utterance Input
            if (row['utterance'] == available_utt[similar_sentence_number]):
                print (chatbot_name, ': ', df['utterance'][index+1])


else:
    print('عذراً، لم أفهم')
```

The Algorithm measures the Similarity between the User Input Text and all of the (input text) Utterances related to this predicted Context using tf-idf and Cosine Similarity. Then it returns the Utterance Response of the most similar Utterance Input.

5. Testing the Chatbot

It was tested using sentences similar to those in the Dataset but not the quite the same ones. The results were good enough to meaning it could predict both the context and Utterance well, as shown in the Chatbot Demo. The Link for the Demo is attached below. Please notes that, any “Hamza” should be added correctly in the sentence while testing to make the Chatbot able to do its prediction.

Reference for Arabic Stemmers Comparison

Almazrua, A., Almazrua, M., & Alkhalifa, H. (2020, December). Comparative Analysis of Nine Arabic Stemmers on Microblog Information Retrieval. In *2020 International Conference on Asian Language Processing (IALP)* (pp. 60-65). IEEE.

Chatbot Demo Link

https://drive.google.com/file/d/1E-O1q4wlCoE8Hf2Gd_FZEQh4VCWRVpGa/view?usp=sharing