****

**ARVIND PAWAR**

**ASHWINI KUMAR PATHAK**

**SHWETA GUPTA**

**SINCHEN GUNDMI**

**VIGNESH THULASI DASS**

**Final Project: Husky Chatbot for Covid-19**

**EAI 6010 Applications of Artificial Intelligence**

**Instructor: Sergiy Shevchenko**

**INTRODUCTION:**

As Covid-19 has been declared a pandemic on 11 March 2020 by WHO (World Health Organization), we have decided to contribute in fighting this disease by utilizing AI (Artificial Intelligence).

**DESCRIPTION OF TECHNOLOGIES USED:**

To provide a complete workflow of chatbot, following steps need to be followed:

**I.** **Socket Programming:**

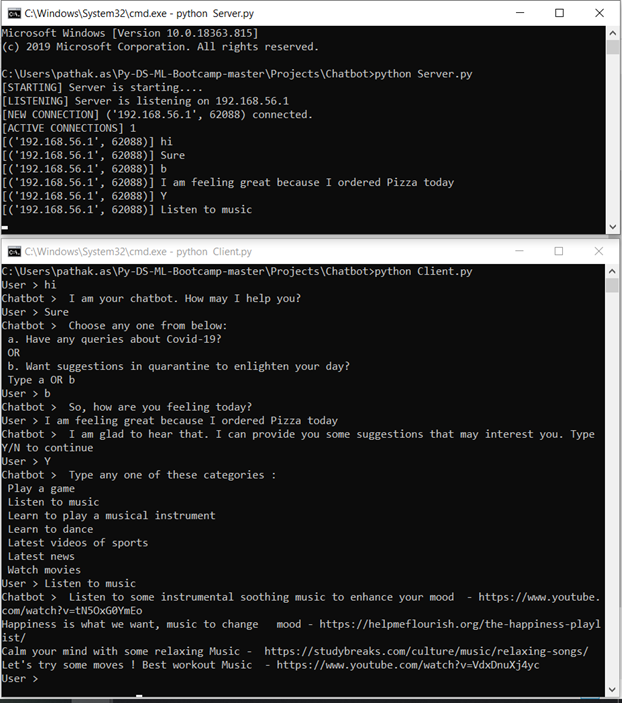
Socket programming is the concept of connecting more than two nodes on a network to communicate with each other [1]. It has been used as a front end to accept user’s questions and provide the best possible answers. It contains a server script which has the capability to handle multiple clients or users who can connect to the server at the same time. Smooth transitioning among major objectives of chatbot has been achieved by developing high performance functions inside socket programming.

To understand how this socket programming works in command prompt, refer to the “working demo” for more information. In this project, socket contains two major scripts as follows:

1. Server script: In server script, it accepts the request from client with how many active connections are present along with client IP address

2. Client script: In the client script, chatbot is answering the user's query based on each question by the user

See below to demonstrate a screenshot of how chatbot will handle concerns of citizens:



Note that chatbot is divided into two major objectives:

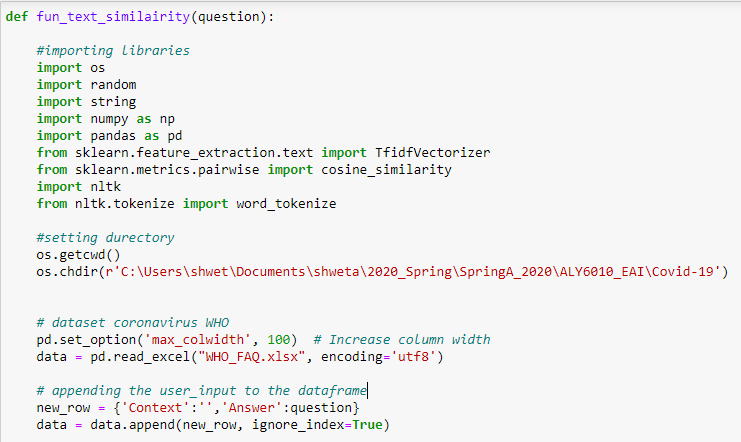
1. To resolve queries about different aspects of COVID-19 like symptoms of COVID-19, suggestions for a pregnant woman, spread of COVID-19 in USA, and many more
2. To provide recommendations to users to improve quarantine life by analyzing their sentiments

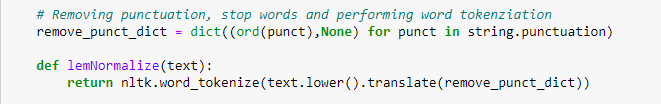
**II.** **Topic Identification:**

**Steps for Topic Identification**

1. **Load the data:** In this step wehave uploaded the excel with general QA about COVID-19.
2. **Text Mining:** In this step we have processed the information to extract valuable information using the following steps

* Converting the text to lower case. This would ensure our similar words with different cases are treated as duplicates.
* Removing Punctuation, Punctuation does not add any value to the text and therefore removing it will not cause loss of information, on the contrary it would help to reduce the length of the text.
* Removing Stop words- These are the most common words (such as “the”, “a”, “an”, “in”) used in sentences, which do not add any information. We have used the predefined list of words defined in the library- stopwords of NLTK package to filter out the common words.
* Word Tokenization- Tokenization is the process of tokenizing or splitting a string, text into a list of tokens and word tokenization is the process of splitting a large sample of text into words.
* To perform all the above steps, we have used NLTK library. Below code represents the implementation of the same.

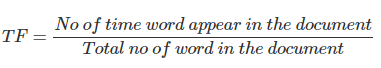




1. **Feature Extraction:** We would be performing text feature extraction which is the process of taking out a list of words from the text data and then transforming them into vectors.

We have used the TF-IDF (Term Frequency – Inverse Document Frequency) NLP technique to extract features from the text. It basically tells the importance of the word in the corpus or dataset. This method is a widely used technique in Information Retrieval and Text Mining.

The first term (TF) is the normalized Term Frequency computed as



and the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

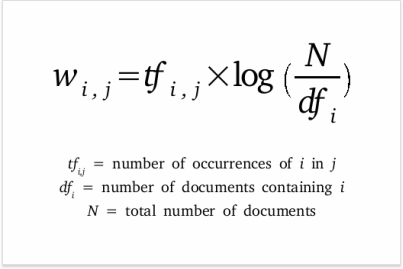
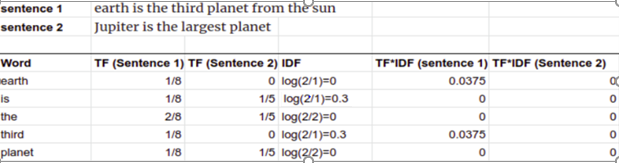
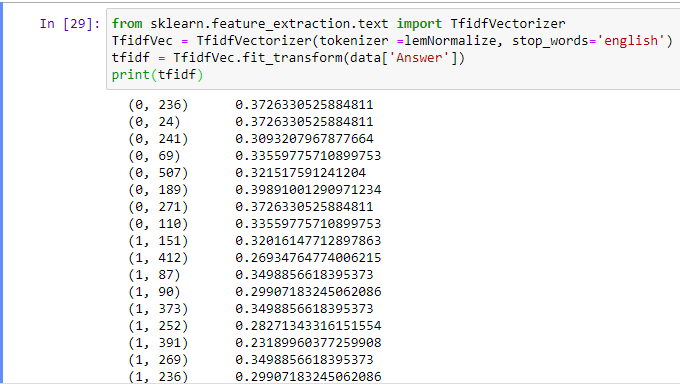


Image source: <https://www.link-assistant.com/images/news/tf-idf-tool-for-seo/screen-03.png>

The below example explains how the it works.



We have used TfidfVectorizer library from sklearn to compute the TF-IDF Vectors at word level. Below is the implementation of the same.



1. **Applying cosine\_similarity**: It measures the cosine of the angle between two vectors. In this case, each document can be presented as a vector whose direction is determined on a set of the TF-IDF values in the space using the below formulae

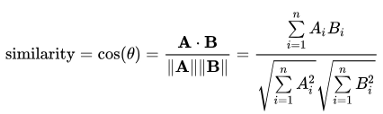
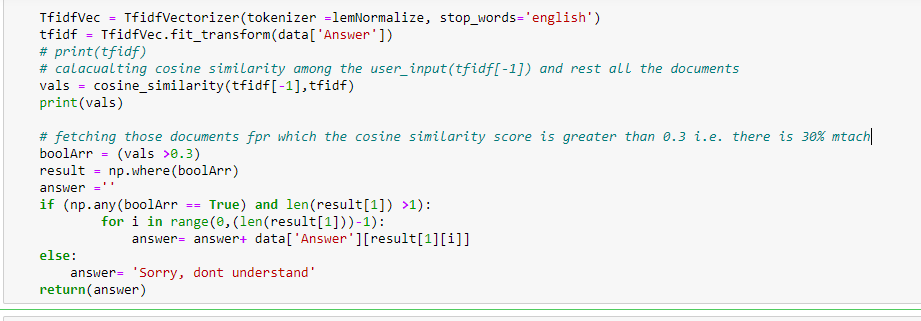
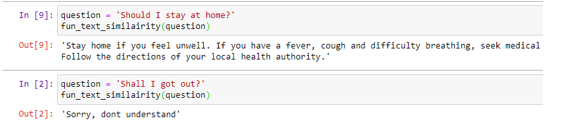


Image Source: https://miro.medium.com/max/852/1\*5J8YlnfnZlzFobQC9cGk-w.png

We have used cosine\_similarity function from sklearn to calculate the cosine similarity value for every single combination of the user input and information available with us. Below code represents the same.



Below is the output for the use question:



Advantages of using cosine similarity:

* The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together.

Disadvantage of using cosine similarity:

* As can be observed from the above output, using cosine similarity we can get the answers by matching the text, but it fails to match the intent of the user input and therefore we have moved ahead with more Pre-trained sentence encoders available with tensforflow\_hub.

1. **Using Tensorflow Hub**: TensorFlow Hub is a repository for finding, publishing, and reusing the part of machine learning modules in TensorFlow. By reusing the module, we can train a model with smaller samples, improve generalization, and speed up the model building process.

From TensorFlow Hub, we used Universal Sentence Encoder (USE) for semantic similarity to match the intent of user input. It was built to support researchers for analyzing natural language texts. It embeds the texts by calculating and plotting the correlation matrix between different terms and successfully learns to capture the meaning of different words

Below is the output of TensorFlow Hub:



From the above result, we can observe that the model has been able to understand and match the intent of user input who wants to know the current status of coronavirus. Unlike cosine similarity, models built using TensorFlow don't require exact same keywords to match the sentence and retrieve the answer. TensorFlow Hub can interpret the synonyms of words, so it returns the appropriate responses for the user’s query.

**III.** **Sentiment Analysis:**

**VADER** (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features which are generally labelled according to their semantic orientation as either positive or negative. This is because VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

Advantages of using VADER

• It works exceedingly well on social media type text, yet readily generalizes to multiple domains.

• It **does not require any training data** but is constructed from a generalizable, valence-based, human-curated gold standard sentiment lexicon.

• It is fast enough to be used online with streaming data.

• It does not severely suffer from a speed-performance tradeoff.

What did not work:

**TextBlob** is a popular Python library for processing textual data. It is built on top of NLTK, another popular Natural Language Processing toolbox for Python. TextBlob uses a sentiment lexicon (consisting of predefined words) to assign scores for each word, which are then averaged out using a weighted average to give an overall sentence sentiment score. Three scores: “polarity”, “subjectivity” and “intensity” are calculated for each word.

What did not work:

While an unsupervised approach is built on specific rules, ideal for generic use, supervised approach is an evolutionary step that is better to analyze large amounts of labeled data for a specific domain. Though unsupervised approach is good for the start, using the open source unsupervised approach will not produce consistent results in domain-specific requirements. In our analysis, we observed while some libraries work better at detecting positive sentiments, others work better with negative data sets.

Benefits of using Vader over TextBlob:

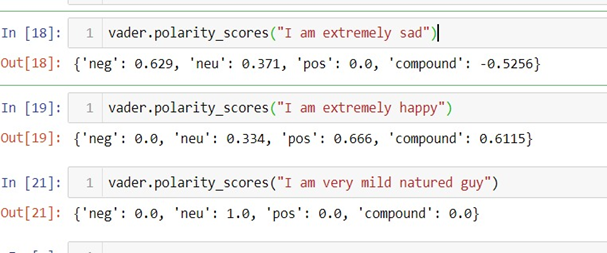
· VADER performs very well with emojis, slangs, and acronyms in sentences.

· The results of VADER analysis are not only remarkable but also very encouraging. The outcomes highlight the tremendous benefits that can be attained using VADER in cases of micro-blogging sites wherein the text data is a complex mix of a variety of text.

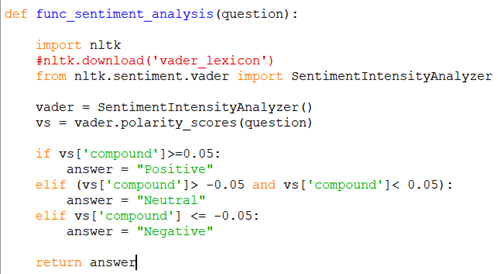
**Example:**

• The Positive, Negative and Neutral scores represent the proportion of text that falls in these categories. This means our sentence was rated as 67% Positive, 33% Neutral and 0% Negative. Hence all these should add up to 1.

• The Compound score is a metric that calculates the sum of all the lexicon ratings which have been normalized between -1(most extreme negative) and +1 (most extreme positive). In the below example, lexicon ratings for and “Best thing” are 2.9 and respectively 1.3. The compound score turns out to be 0.63, denoting a very high positive sentiment.



**Snapshot of code:**

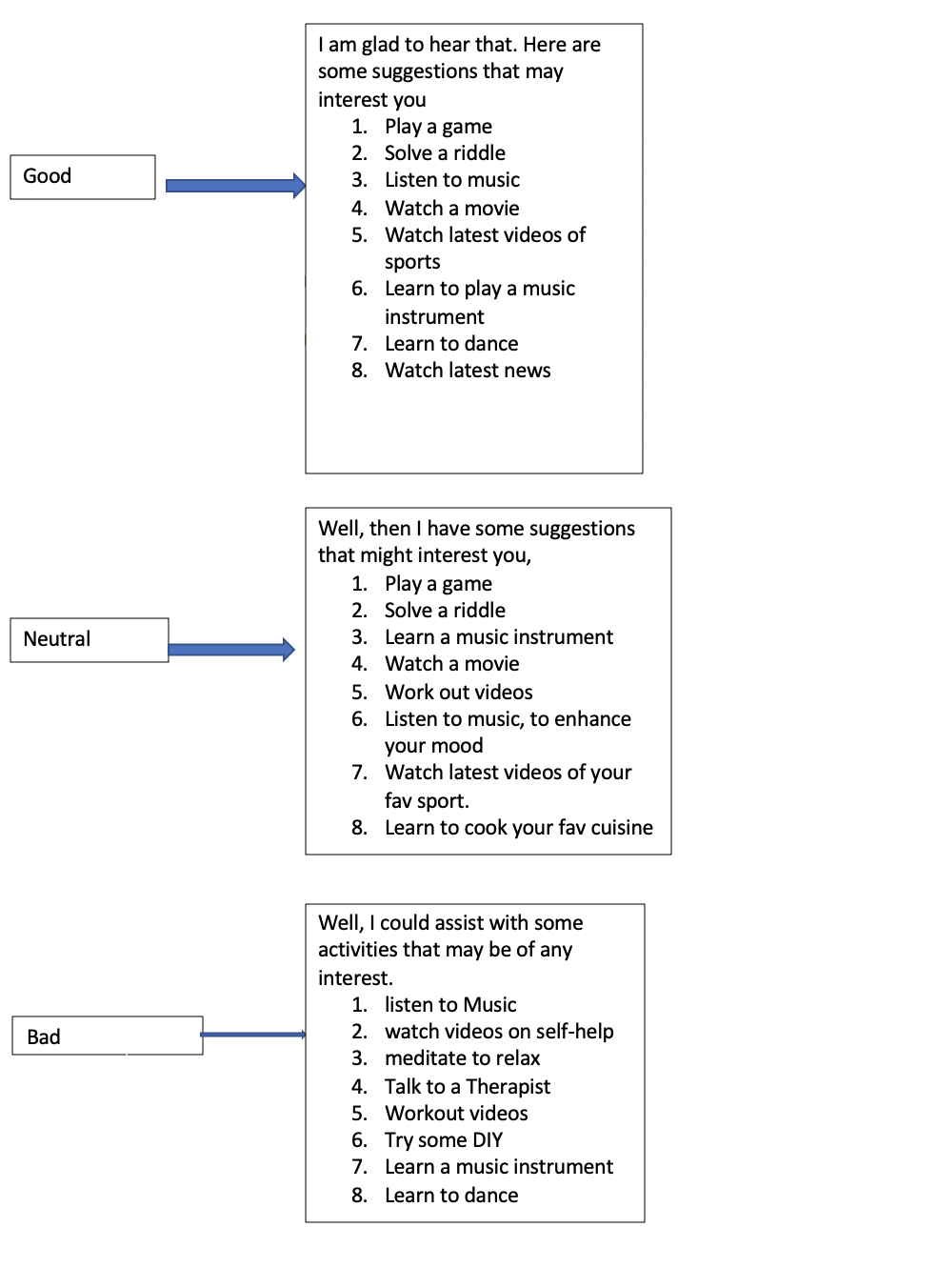


**IV**. **Recommendation**

Social distance has been very challenging during the COVID-19, technology has united all of us during this time. When the user inputs a word on the chatbot, we are analyzing the sentiment using the cosine similarity. This is happening when the user chooses the option b on the queries section.

After analyzing the sentiment, we are using the VADER function to classify the sentiment into positive, neutral and negative, and based on each of the classified sentiment, we are providing recommendations. According to a study most of the people are spending time on social media, music and on art during the isolation, and our recommendations are based on it.

As mental health and physical health matter the most during the isolation, we have suggested certain ways of help. The Figure below explains the recommendation for each sentiment.

****

**CHALLENGES:**

In developing chatbot for resolving the concerns, following challenges occur during the project:

1. Tensorflow\_text has version compatibility issues with Windows; hence we could not implement Universal Sentence Encoder (USE) on Windows system
2. Using cosine similarity, we can match the similar words in two different sentences, but we cannot find the similarity in the intent of the two
3. After merging the separate codes for topic identification, cosine similarity and sentiment analysis in socket programming, integration failed at many instances

**SCOPE:**

While building a chatbot, there is always a room for improvement. Hence, following are the various scope for this chatbot:

● Front-end can be made attractive and appealing to the user

● Sentiment analysis can contain more classification to better understand the emotions of user

● More sophisticated system can be built to give recommendations to user

**CONCLUSION:**

After the all above analysis of chatbot, we have concluded and learning following aspects of building a chatbot:

1. The Tensorflow-hub has been most effective in topic identification as compared to cosine similarity function.
2. The chatbot can answer the questions asked in 16 different languages, which is one of the advantages of the USE module.
3. Socket programming acts as easy and effective in connecting more than 1 client to the server and has the capability to handle long texts.
4. The recommendations on the chatbot are based on the sentiment and these recommendations can uplift the mood during the social distancing.

**REFERENCES:**

[1] Socket Programming in Python. Retrieved from<https://www.geeksforgeeks.org/socket-programming-python/>

[2] Sentiment Analysis using VADER. Retrieved from <https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f>