DATASET :

HP1.txt = Harry Potter and the soccer's stone

NAMO1.txt = US SPEECH by Narendra Modi

NAMO_gst.txt = Speech on GST by Narendra Modi

Our project is structured as below:

Markov

- **Markov.py** (The below commands are there in markov.py file also as comments)

    - **python3 markov.py -w database/w3_small.db -n 3 --test --coca --predict a baby ----------> will give the ordered dictonary for the next word with the word count**
    - **python3 markov.py -w database/big/w3.db -n 3 -words 100 --predict are you ----------> will give the ordered dictionary along with the predicted paragraph of 100 words with w3.db as the database and -n 3 tells that you want the third word after giving "a baby" as input.**
    - We can use **python3 markov.py -w database/big/w4.db -n 4 -words 100 --predict are you** to predict the fourth word
    - Sometimes though you give 100 words to predict , it only predicts a few, it's because , since its a store and predict model, if no word is found in database it gives empty output.
    - Execution is also shown in results in ppt.

```
        parser = argparse.ArgumentParser(description = 'usage %prog ' + '-m<model>/-d<dataset> -n<n-gram> ')
        parser.add_argument('-w', dest='dataset', type = str, action = 'store', help='Weights to train')
        parser.add_argument('-f', dest='data_file', type = str, action = 'store', help='Text file to train/test')
        parser.add_argument('--test', action='store_true')
        parser.add_argument('--train', action='store_true')
        parser.add_argument('--coca', action='store_true')
        parser.add_argument('-m', dest='model',  type = str, action = 'store', help='Trained model')
        parser.add_argument('-n', dest='n',  type = int, action = 'store', help='N in N-gram')
        parser.add_argument('-words', dest='words',  type = int, action = 'store', help='Next n words to generate')
        parser.add_argument('--predict', nargs = "*", dest = 'predict', action='append')

        options = parser.parse_args()

        # if(options.test):
                # # python3 markov.py -m database/HP1.db -f dataset/HP7.txt --test -n 3
                # m = MarkovChain(options.model, options.data_file, options.n)
                # print(m.validate())

        # if(options.train):
                # m = MarkovChain(options.dataset, options.data_file, options.n)
                # m.learn_from_text()
```

- **Server.py**
- **Client.py**
- **Readme.md**
- **database (folder)**
    - **SQL Database created with wikipeida corpus for testing**

- **dataset(folder)**
  - **big(folder) has the coca n-gram dataset**
  - **spoken(folder) has the spoken english dataset**
- **Clt.py**
- **Store.py** -> will store all the sql .db files
- **File_process.py** -> will process all the text dataset files

· **char_by_char**:
- History5.p -> saved model
- Model5.h5 -> saved weights
- Graphs for accuracy and loss
- rnn4.py which was used to train the model
- Testing jupyter notebook to performing testing
- Dataset HP2.txt

· **word_by_word**:

○ **Harry Potter**: Input File: HP1.txt
Adam:*****
RMSprop:****

○ **US Speech**: Input File: NAMO1.txt
- **3 word** *****
- **Sentence** *****

○ **Speech on GST**: Input File: NAMO_GST.txt
- **3 word** *****
- **Sentence** *****

NOTE:
- Each folder in word by word folder tells the dataset first and then inner folder names the approach.
- Before executing the test notebooks, please check the file path of dataset.
- Every folder has its required dataset text file embedded. Its not mentioned in the project file structure.
- ONLY USE THE TEST FILE IN ALL FOLDERS.

Each folder with ***** contains
- .ipynb: -> original file
- .h5, .json: -> Saved Model
- Graphs for accuracy and loss
- Predicted text
- Test file to perform any test cases
  - Every test notebook has its particular statement like: print(generate_seq(loaded_model, tokenizer, 2, 'because her sister and', 15)) where the **last parameter** is the **number of words to predict**.
  - Please use the particular code line as shown in test notebook to test using generate_seq function.

Note : Do not perform any changes in the original jupyter notebook , as it requires to train the model again. Any testing should be performed in the test.ipynb file.