ReadME for Assignment 1 (Intro to NLP)

How to execute files

1. **Tokenizer**

- To get the tokenized output for the given corpus, the command should be

```
'python3 tokenizer.py <corpus_path>'
```

- The output is a list of lists, in which each list contains the sentence tokenized into words with appropriate place holders for URLs, Hashtags, Mentions, Numbers and Mail IDs.

2. **Language Model**

- For this file the command should be

```
'python3 language_model.py <lm_type> <corpus_path>'
```

- The lm_type is
  - i for Linear Interpolation
  - g for Good Turing
- The output generated contains 4 text files conating the perplexities of each sentence from the corpus divided into training and testing set and a prompt asking for an input sentence.
- After entering the input sentence, the model calculates the probablity score of that sentence using the lm_type earlier specified by the user.

3. **Generation**

- The command for running this file is

```
'python3 generator.py <lm_type> <corpus_path> k'
```

- The lm_type is
  - i for Linear Interpolation
  - g for Good Turing
  - n for Normal N-Gram Model (without any smoothing techniques used)
- k denotes the number of candidates for the next word to be printed.
- The output is a list of k words (in decreasing order of probablities) along with their probablity scores printed.

Assumptions

1. **Tokenizer**

- For few abbrevations like Mr./Mrs./Dr. which were present in the given dataset, the tokenizer did not split the string into sentences, however for other abbrevations it may do so.
- The URLs with domain names as http/https/www are the only ones which can be identified by the tokenizer.
- Decimal Numbers, Integers and Amount(Numbers preceeded by $) are identified as numbers.

- Punctuation Marks are only identified by the tokenizer and not removed in tokenizer.py, however as we proceed to language models and next word genrators, the punctuation marks are removed.
- Tokenizer does not convert the text into lower case.
- Dates, percentages, and other kind of numerical values are not identified by the tokenizer.
- The tokenizer returns a list of lists, each list containing a tokenized sentence with replaced placeholder for required tokens.

2. **Language Model**

- For a given corpus the language model splits it into the test, train and held out data. It generates 4 text files contaning the Average Perplexities and perplexity of each sentence present in the testing and the training data.
- The model generates a output prompt asking for an input sentence and calculate it's probablity score on the lm_type choosen by the user.
- The trigrams for which the context word(bigram) was not present in the corpus, is assigned a very low probablity which is equal to 1/(total no. of trigrams)

3. **Generation**

- For the N-Gram model generation (without the use of any smoothing techniques) the value of N can be decided by the user (the context words could be trigrams, bigrams, 4grams etc.)
- If the given context is not already present in the data, them the model predicts words on a random basis with weight given to the frequency of words in the corpus.
- If k is more than the number of words predicted, then we output a list of probable next words less than k.
- For Next Word Prediction Task using Linear Interpolation and Good Turing only trigram lm is used along with the choosen smoothing method.
- If the context words are not present, the model does not print any possible next words in case of Good Turing Smoothing.