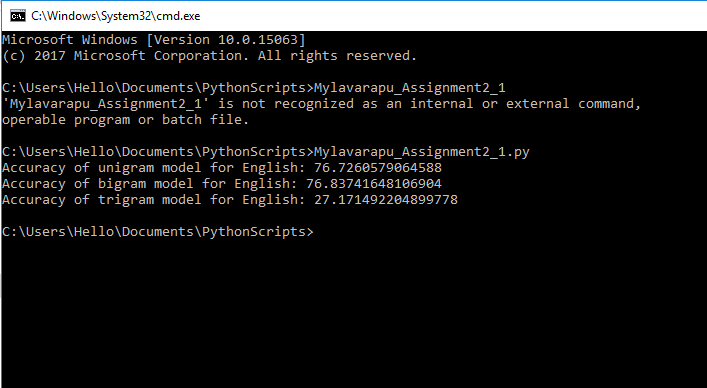
**REPORT**

Note:- There are two .py files for two questions. They don’t read any input files or parameters.

**PROBLEM 1:**

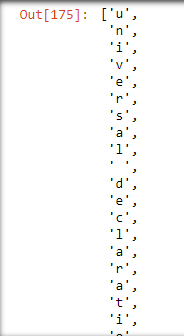
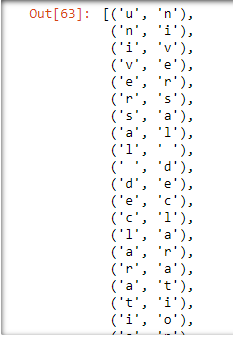
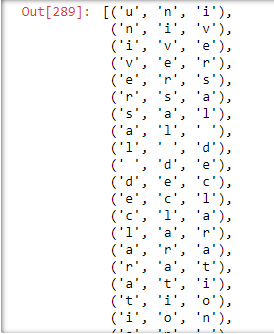
The output for first file ‘Mylavarapu\_Assignment2\_1.py’ is as follows:-



**Procedure:-**

In order to achieve the accuracy, I followed the below procedure:-

1. Created Train, Dev and test by splitting the given dataset.
2. Then I created the Unigram, Bigram and Trigram models for all the four languages using ‘**FreqDist’** and ‘**ConditionalFreqDist’**, along with the standard nltk library ‘**ngrams’**. Initially the given train data in the form of string is split into individual characters and then an ngram model is built upon that.

1. Once the models are ready, I’ve processed the English TEST dataset – the test dataset is given in the form of ‘nltk.collections.LazySubsequence’. So, it is converted to text, punctuations and numbers are removed from it.
2. **English V/s French Unigram Model**:-

* In case of unigram model, probability is calculated simply by dividing the frequency of each character encountered by the total number of characters in the train corpus.
* The same process is followed for the French unigram model as well. For each word, probability is calculated and if the probability of English model is greater than French model, ‘English’ is predicted and vice-versa.
* Finally based on the count of the number of English words, accuracy is determined as 76.72%.

1. **English V/s French Bigram Model:-**

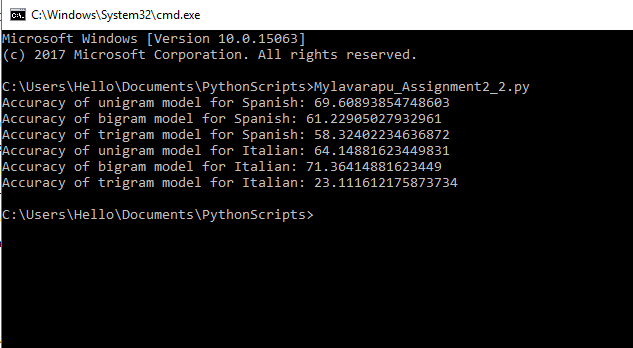
* The total characters are split into two categories: For the first character of every word, probability is calculated by considering the previous character as ‘ ’(space). Hence the probability of each letter would be the count of the occurrence of a space followed by that letter divided by the count of the occurrences of space.
* On the other hand, for all other characters, probability of the occurrence of that letter given the previous letter is considered.
* Again, the same process is followed for the French model as well. For each word, probability is calculated and if the probability of English model is greater than French model, ‘English’ is predicted and vice-versa.
* In this case also, the overall accuracy turned out to be approximately 76.83%.

1. **English V/s French Trigram Model:-**

* The total characters are split into three categories in this case: For the first character of every word, probability is automatically considered as 1. Because we cannot calculate the probability of its two previous words.
* For the second character of every word, probability is calculated by considering the first character as ‘ ’(space). Hence the probability of each letter would be the count of the occurrence of a space followed by previous character and that character divided by the count of the occurrences of space and previous character.
* On the other hand, for all remaining characters, probability of the occurrence of that character given the previous two characters is considered.
* Again, the same process is followed for the French model as well. For each word, probability is calculated and if the probability of English model is greater than French model, ‘English’ is predicted and vice-versa.
* In this case, as it is very difficult to find the three group of characters from a small train dataset of 1000 characters, the overall accuracy decreased to approximately 27.17%.

**PROBLEM 2:**

The output for the second file ‘Mylavarapu\_Assignment2\_2.py’ would be as follows:



**Procedure:-**

The same procedure is followed in this case as well, except the fact that a total of 6 evaluations are made. First three models contain Unigram, bigram and trigram based on Spanish Test data whereas the other three contains Unigram, bigram and trigram based on Italian test data.

Below are the accuracy details obtained for the six models:

|  |  |  |
| --- | --- | --- |
| **Model** | **SPANISH TEST DATA** | **ITALIAN TEST DATA** |
| **Unigram** | 69.6 | 64.14 |
| **Bigram** | 61.22 | 71.36 |
| **Trigram** | 58.32 | 23.11 |

From the above table, it is clear that Spanish has more chances of getting distinguished easily than Italian.

**Challenges Faced: -**

* The main difficulty was to calculate the frequency distributions of Trigrams. Because the ‘FreqDist’ and ‘ConditionalFreqDist’ doesn’t work with the Trigrams.
* So I spent some time in finding out a way for this. But I finally used ‘Dictionary’ in python to put all the trigrams as ‘keys’ and the frequency values as ‘values’.
* Hence it became easy for me to search for the frequency values as it is just a ‘key-value’ pair.