ITCS 5111- Natural Language Processing
Language Modelling Report
Archit Parnami

Objective:

Build Unigram, Bigram and Trigram models for given English, French, Spanish and Italian text corpora and use them to predict the language of input text.

1. Data Preprocessing

- For each language
  - Strip the text for any whitespace at the beginning or in the end.
  - Remove punctuations
  - Convert the text to lowercase.
  - Replace any new line characters with underscore ("_")
  - Replace whitespace with underscore ("_")
- Partition the data for each language into training, development and test sets.

2. Building the N-Gram Model

- Using the nltk.util.ngrams() function find the Unigrams, Bigrams and Trigrams from each language text.
- Find the frequency distribution of each of the Unigrams, Bigrams and Trigrams using the nltk.FreqDist() function.
- Write a function named "compute_pdf" for computing probability distribution from frequency distribution.
- Find the probability distribution of Unigrams, Bigrams and Trigrams.

3. Finding probability of given test word

- Unigram Probability Calculation
  - Pu("ABC")
    = Pu(A) * Pu(B) * Pu(C)
  - Where Pu is Unigram probability

- Bigram Probability Calculation
  - Pb("ABC")
    = P(A/_) * P(B/A) * P(C/B) * P(_/C)
    = (Pb(_A) / Pu(_)) * (Pb(AB) / Pu(A)) * (Pb(BC) / Pu(B)) * (Pb(C_) / Pu(C))
  - Where Pb is the Bigram probability

- Trigram Probability Calculation
  - Pt("ABC")
    = P(B/_A) * P(C/AB) * P(_/BC)
    = (Pt(_AB) / P(_A)) * (Pt(ABC) / P(AB)) * (Pt(BC_) / P(BC))
  - Where Pt is the trigram probability

Points to note:

- New lines in the raw data were replaced by underscore ('_'). This was done because we are not aware about the position of the words in test set. The words in test may or may not be at the begin or end of a sentence.
- For trigram probability only single padding is done on both the ends and double padding is ignored.

4. Predicting the language

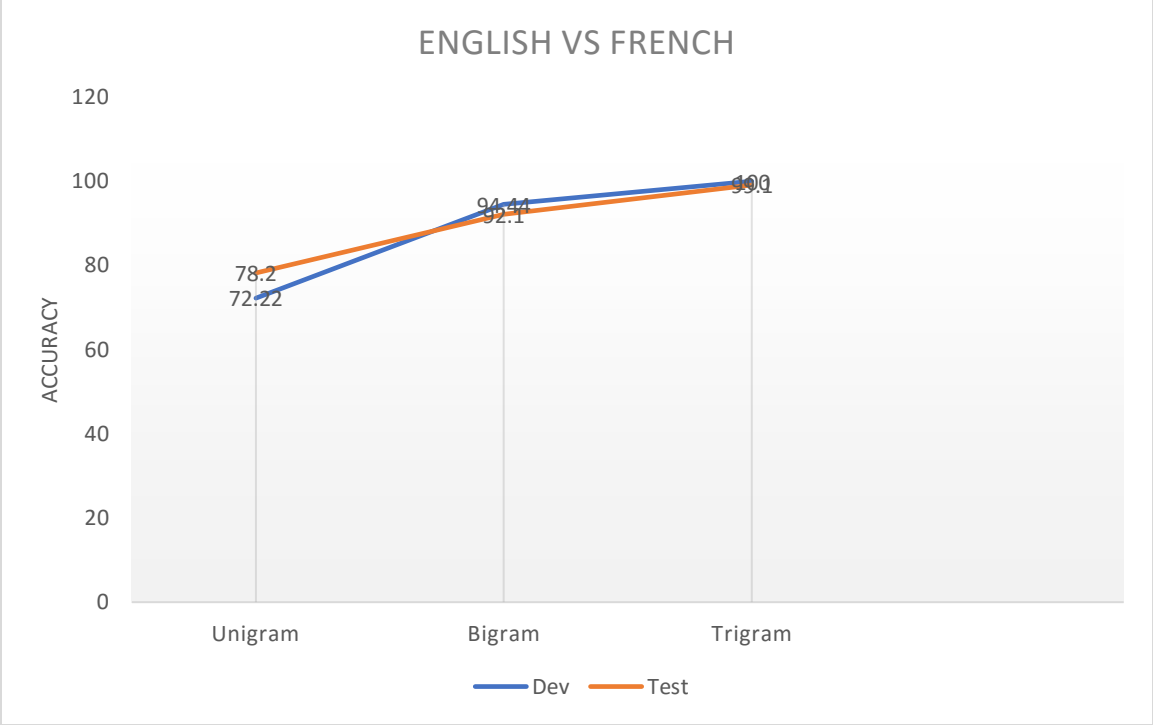
- For each word in the test set
  - Compute the Unigram probability of the word for language 1 and for language 2.
  - Increment the count for language 1 matches **when probability(languag1) >= probability(language2)**
  - Do the same for Bigram and Trigram models.

5. Results

- Test Set: English, Language 1: English, Language 2: French

Prediction Accuracy for English

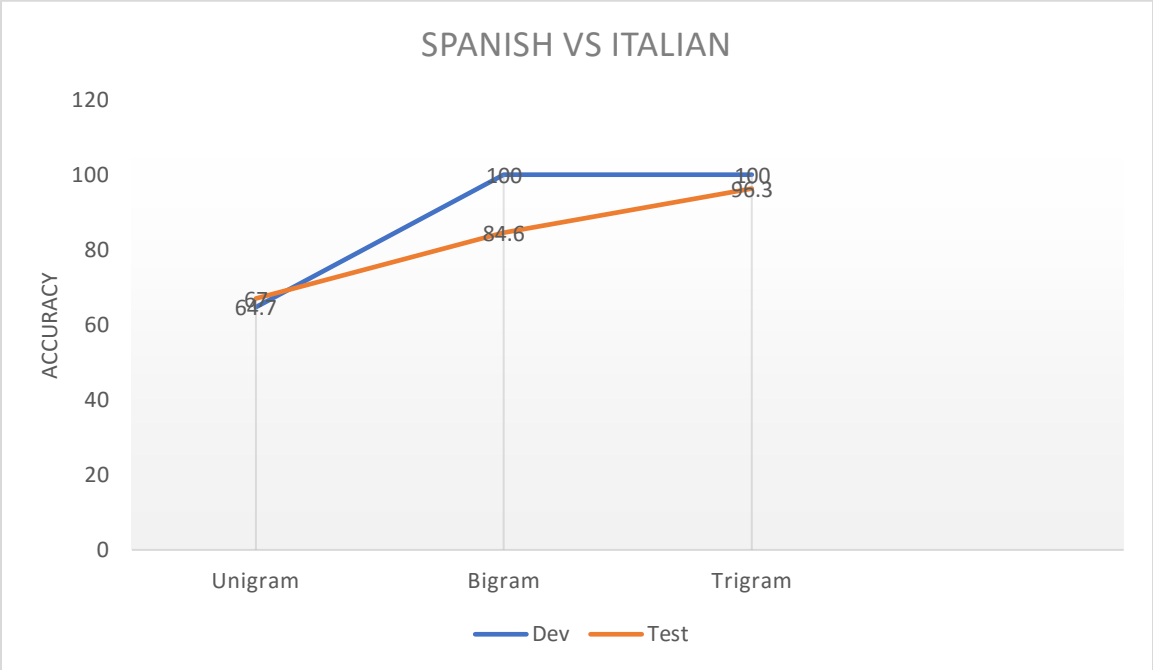
	DEV	TEST
UNIGRAM	72.22	78.20
BIGRAM	94.44	92.10
TRIGRAM	100.00	99.10



- Test Set: Spanish, Language 1: Spanish, Language 2: Italian

Prediction Accuracy for Spanish

	DEV	TEST
UNIGRAM	64.70	67.00
BIGRAM	100.00	84.60
TRIGRAM	100.00	96.30



6. Analysis

- Accuracy increase with N in N-Gram models.

7. Challenges

- When building the model, it was important to perform data preprocessing to achieve higher accuracy.
- Use of padding to account for extreme end of words was important to get correct probabilities.

8. Output files

- Executing the python program generates 4 text files which have the Unigram, Bigram and Trigram probabilities for each comparison language.