Questions

Table of Contents

1. Question 1 :

(b) - 1000 Short Explanation: Natural Languages such as English follow a Zipfian distribution, which means the frequency of occurrence of any word is inversely proportional to its rank on the frequency table. Hence, we need to solve a simple equation (which we did in the python code attached). The answer is around 336 (assuming there are 200000 words in English), which gives 1000 as the upper bound. (code shared in folder)

2. Question 2 : Regex

Code in datedetecter.py

3. Question 3 : Party or No Party

Here we used levensteine algorithm to match every word in the sentence with the word "party". This way we coul also have fuzzy matching

4. Question 4 : N-grams and hyphenation

For the ngrams we removed the punctuation. The n-grams were generated from a simple for loop. However we could also use the nltk library to do this more efficiently. The hyphenation was done by using the logic that any adverb/adjective followed by n-nouns were hyphenated.

5. Question 5 : Tagging and removal of Stopwords

For this we used the NLTK library. The text was first cleaned and all of its contractions were removed using our custom code.

6. Question 6 : Decision Tree

We used used the following features to make our prediction:

- Avg sentence length

- Avg no. of nouns, pronouns, adjectives…
- Avg no. of common words (inlc stopwords) like "a", "in","the"
- Avg no. of commas

The features were averaged over sentences and each sentence was used as a sample. To make the final prediction, we averaged over the sentences. The decision tree used is implemented in scikit and very much similar to the C4.5 On our run the result was "HP lovecraft" The tree is visualised in iris.pdf