Assignment 3 : NLP

Name: Shreya Sharma

Roll No.: 2015096

Question 1 & 2:

Length of sentence by User Input

1. Tokenized the document into sentences using PunktSentenceTokenizer.
2. Tokenized all the sentences into words (removed all non-alphanumeric characters (even '_')) and hence created the corpus which has the following structure:
    i. Corpus = [[w11,w12,...w1n],[w21,w22,..w2n],...] where wij = jth word in ith sentence

   *note that "" is also as a word and it denotes the start of a sentence.
3. Calculated the probabilities for all n-Grams:
    a. Counted the frequency of each word in the corpus and calculated the probabilities for unigram model using the formula P(w) = C(w)/total_words
    b. Counted the frequency of each pair of words in the corpus and calculated the probabilities for bigram model using the formula P(wi/wi-1) = C(wi-1,wi)/C(wi-1)
    c. Counted the frequency of each pair of words in the corpus and calculated the probabilities for bigram model using the formula P(wi/wi-2,wi-1) = C(wi-2,wi-1,wi)/C(wi-2,wi-1)
4. Generated sentences for all n-Grams:
    a. Chose the first word randomly from the top 5 words and then choose the next most probable word after the previous word from the unigram prob list.
    b. Similar to the above idea. Generate the first word randomly from top 5 of the list of bigrams starting with prev word = "". Then update prev as the current word and keep repeatedly choose next word from 5 most probable next words. Whenever prev is not in the bigram corpus, generate a word using bigram with sentence length = 1 and update prev as this word.
    c. Generate a word using bigram (sentence length = 1) then make prev = "" + generated word. Now generate new words using prev and keep updating it after every new addition of word. We add a word randomly from the top 5 probable trigrams.Whenever a prev is not in the trigram corpus, generate a new one using bigram with sentence length = 1 as 2nd word and prev's 2nd word as 1st word. Keep doing it till a prev is found in the trigram corpus

Sample sentences generated (length = 10):
Comp.graphics:
unigram sentence: *to a of and i is for in it you.*
bigram sentences: *if anyone know of the following is that the image.*
i have to a good book and i am not.
trigram sentences: *if your viewing software on a sun sparcstation x11r4 sunos.*
it has some pixel resampling functions not in the same.
Rec.motorcycles:
unigram sentence: *a i to and of in it you is that.*
bigram sentences: *the road as a bike is a motorcycle club with.*
newsgroups recmotorcyclesharley subject how many times with my opinions wanted.
trigram sentences: *the only thing i still increase my visibility to any.*
i m going and would incourage other club members to.

Question 3.1:

1. Created corpora for both classes using same technique above.
2. Generated unigram frequencies from both corpora
3. Generated bigram probabilities for both classes using laplace smoothing formula :
   $P(wi-1/wi) = (C(wi-1,wi)+1)/(C(wi-1)+|V|)$
4. Take Sentence or Doc file from user
5. Preprocessed user input (sentences+words) similar to how corpora were made.
6. Then word by word added the log probabilities of each word as in naive bayes
   $P(c/x) = [P(x/c)P(c)]/P(x)$
   $P(c/x)$: posterior
   $P(x/c)$: likelihood
   $P(x)$: predictor prior (ignore since common in all terms)
   $P(c)$: prior (assume same for both classes)
   Therefore, $P(c/x) \approx P(x/c)$
   $P(x/c) = P(x1,x2,...xn/c) = P(x1)*P(x2/x1)*P(x3/x2)....P(xn/xn-1)$
   $\log(P(x/c)) = \log(P(x1))+\log(P(x2/x1))+....\log(P(xn/xn-1))$
7. Assign the class whose log probability is higher
8. If a new word occurs, $\log(P(xj/c)) = \log(1/(c(xj-1)+V))$
9. If xj-1 is not present then $\log(P(xj/c)) = \log(1/total\_words \text{ in corpus } +V)$

Question 3.2:

10. Created corpora for both classes using same technique above. Also removed the stop words.
11. Generated unigram frequencies from both corpora and word with freq = 1 is considered “unk” ie <UNK>
12. Generated bigram probabilities for both classes using laplace smoothing formula : $P(wi\text{-}1/wi) = (C(wi\text{-}1,wi)+1)/(C(wi\text{-}1)+|V|)$ and if the word is not present in unigram then it is considered as “unk”
13. Take Sentence or Doc file from user
14. Preprocessed user input (sentences+words) similar to how corpora were made. Also removed the stop words.
15. Then word by word added the log probabilities of each word as in naive bayes and if a word doesn’t appear in unigram then it is considered as “unk”.

    $P(c/x) = [P(x/c)P(c)]/P(x)$

    $P(c/x)$: posterior

    $P(x/c)$: likelihood

    $P(x)$: predictor prior (ignore since common in all terms)

    $P(c)$: prior (assume same for both classes)

    Therefore, $P(c/x) \approx P(x/c)$

    $P(x/c) = P(x1,x2,...xn/c) = P(x1)*P(x2/x1)*P(x3/x2)....P(xn/xn\text{-}1)$

    $\log(P(x/c)) = \log(P(x1))+\log(P(x2/x1))+....\log(P(xn/xn\text{-}1))$

16. Assign the class whose log probability is higher