CSCI 404 Natural Language Processing

Winter 2018 - Homework III

Haley Beavers & Robert Harley

Task 1:
For task 1, we successfully inserted the start and end tokens (<s>, <s/>) into the our sentences and replaced all words that occurred only once in our training copus with the token <UNK>. This was done by first inserting the start and end tokens, followed by getting the frequency of the occurrence of our unigrams, and finally removing the tokens with a single occurrence and replacing them with <UNK>. The sizes of our vocabulary, and the number of bigrams and trigrams were reported in output.txt.

Task 2:
In order to generate sentences for unigrams, we find the most common 30 unigrams and find continue to insert those tokens at the end of the sentence until the end token is added or the word limit for sentences is hit. For our generators we used a word limit of 15.

To generate bigrams and trigrams, we would generate a list of candidate tokens to use as our next word in the sentence by considering the past tokens in the sentence. Once we had a candidate list we picked a random token from the candidate list and used that token as a means of deciding the next word in the sentence. This is done until an end of token is placed in the sentence or the sentence limit is met.

Once the sentences are generated, we use the chain rule of probability in order calculate the probability of that sentence being generated based on the training corpus. This was done by summing the probabilities in log-space using the equation below.

$$P(W) = P(w_1, w_2, ..., w_n) = \Sigma^{n}{}_{i=1} \ logP(w_i|w_1...w_{i-1})$$

Task 3:
The perplexity was calculated using the equation below on our training corpus. There is an error with the perplexity of the trigrams that we could not find the bug in, but we believe that the perplexity for the unigram and bigram are correct.

Unigram:
$$Perplexity(W) = exp(-\frac{1}{N}\Sigma^{N}{}_{i=1} logP(w_i))$$
Bigram:
$$Perplexity(W) = exp(-\frac{1}{N}\Sigma^{N}{}_{i=1} logP(w_i|w_{i-1}))$$
Trigram:
$$Perplexity(W) = exp(-\frac{1}{N}\Sigma^{N}{}_{i=1} logP(w_i|w_{i-2}w_{i-1}))$$
A summary of our functions can be found below.

Summary of functions:

***file_text(filename)*:** Adding all lines in file to a list of strings
***add_sym(corpus)*:** Adds <s>, </s> at beginning and end of sentences replaces all tokens with count of one with <UNK>
next_uni(tokens): Randomly picks the next token to use in the sentence from the given list
log_prob(sentence, words): Computes log probabilities of generated sentence
generate_uni(unigrams, words): Constructs sentence using unigrams
next_bi(prev_token, unigrams, bigrams): Finds most probable bigram given previous token
generate_bi(unigrams, bigrams): Constructs sentence using bigram probabilities
first_tri(trigrams): Finds most probable bigram given previous token
next_tri(prev_tokens, trigrams): Finds most probable bigram given previous token
tri_sentence(trigrams): Constructs sentence using unigrams
def generate_tri(unigrams, bigrams, trigrams): Constructs sentence using trigrams
perplexity(n_grams, words, k): Computes complexity for a unigram, bigram, or trigram given for a k value of 0, 1, 2 respectively