NLP HW 1

Name: Vincent Lin (Zhenye)

Late Day: 1 remaining

Written

Note: for all expressions of $< s >$ and $< /s >$ there are no spaces in betwee, I had to adjusted it b/c of latex.

1. The resulting expression is P($< s >$)× P(I | $< s >$) × P(do | $< s >$ I) × P(not | $< s >$ I do) × P(like | $< s >$ I do not) × P(eggs | $< s >$ I do not like) × P(and | $< s >$ I do not like eggs) × P(Sam | $< s >$ I do not like eggs and) ×P($< /s >$ | $< s >$ I do not like eggs and Sam)

2. P($< s >$)× P(I | $< s >$) × P(do | I) × P(not | do) × P(like | not) × P(eggs | like) × P(and | eggs) × P(Sam | and) ×P($< /s >$ | Sam).

3. P($< s >$)× P(I | $< s >$) × P(do | I) × P(like | do) × P(eggs | like) × P($< /s >$ | eggs)
Without Laplace Smoothing, the probability of this expression is 0 since P(like | do) and P($< /s >$ | eggs) = 0.
With Laplace Smoothing, the probability of this expression will be nonzero. Since there is 10 unique tokens (including the starting and the ending token), we add one to the numerator of the and 10 to the bottom of the dominator of Laplace smoothing.We get the equation $P_{Laplace}(w_i) = \frac{c_i+1}{N+10}$ to calculate each of the probability term.
P($< s >$)× P(I | $< s >$) × P(do | I) × P(like | do) × P(eggs | like) × P($< /s >$ | eggs)
= $1 \times 4/14 \times 2/14 \times 1/11 \times 2/11 \times 1/11 = .00006133182$

Programming

1. I used a dictionary that maps the count and the word together. Laplace Smoothing requires us to add one to every single count. However, I do that later when I calculate the probability. So the Count without Laplace Smoothing is:

Unigram:[('$< /s >$', 4), ('$< s >$', 4), ('i', 4), ('sam', 4), ('am', 3), ('not', 1), ('eggs', 1), ('like', 1), ('do', 1), ('and', 1)]

Bigram:[('$< s >$ i', 3), ('$< /s >< s >$', 3), ('sam $< /s >$', 3), ('am sam', 2), ('i am', 2), ('$< s >$ sam', 1), ('eggs and', 1), ('i do', 1), ('am i', 1), ('like eggs', 1)]

The Count with Laplace Smoothing is:

Unigram:[('< /s >', 5), ('< s >', 5), ('i', 5), ('sam', 5), ('am', 4), ('not', 2), ('eggs', 2), ('like', 2), ('do', 2), ('and', 2)]

Bigram:[('< s > i', 4), ('< /s >< s >', 4), ('sam < /s >', 4), ('am sam', 3), ('i am', 3), ('< s > sam', 2), ('eggs and', 2), ('i do', 2), ('am i', 2), ('like eggs', 2)]

2. I used Laplace smoothing technique to calculate all the bigrams and the unigram probability and stored them in a dictionary.

Note: the probability for the bigram can be read as the probability of the second word given the first word.

Unigram: [('< s >', 0.14705882352941177), ('< /s >', 0.14705882352941177), ('i', 0.14705882352941177), ('sam', 0.14705882352941177), ('am', 0.11764705882352941), ('eggs', 0.058823529411764705), ('like', 0.058823529411764
('not', 0.058823529411764705), ('do', 0.058823529411764705), ('and', 0.058823529411764705)]//

Bigram: [('< s > i', 0.2857142857142857), ('< /s >< s >', 0.2857142857142857), ('sam ¡/s¿', 0.2857142857142857), ('am sam', 0.23076923076923078), ('i am', 0.21428571428571427), ('eggs and', 0.18181818181818182), ('like eggs', 0.18181818181818182), ('not like', 0.18181818181818182), ('do not', 0.18181818181818182), ('and sam', 0.18181818181818182)]

3.I used the function from 2 and 1 to train the movie scrip to yield the following bigram model. I perform any preprocessing of lowercasing everything and did not remove any punctuation since I think I should include "Dr." or "it's" in the final table.

the results are the following:
('< /s >< s >', 0.25186379206125326), ('. < /s >', 0.21979036801034307), ('. .', 0.04040264117837189), ('of the', 0.03605050393177539), ('< s > the', 0.034354221237154946), (', but', 0.02580999450851181), ('< s > a', 0.025236751964537577), (', and', 0.024561929010034445), ('in the', 0.020678652512732404), ('is a', 0.017361948396431154), (', the', 0.014277869302580999), ('the film', 0.013267326732673267), ('of a', 0.012681360062022373), ('to the', 0.012221634643633586), ('to be', 0.011634056054997356), ('and the', 0.011082174322602095), ('in a', 0.010799533656501197), ("< s > it's", 0.010729397541809389), ('it is', 0.009522031366691561), ('the movie', 0.009405940594059406)

4. For number 4, I used Laplace Smoothing technique to generate the n-gram up to 5. I had to use '$' to represent the beginning of the sentence and '#' to represent the end of the sentence since NLTK parses the token '< s >' as '<', 's', and '>'. The preprocessing I did for this lab is to lower case every words after being processed by the sentence tokenizer and as with 2 I did not remove any punctuations.For postprocessing, I added "." to each sentence

The result are as follows:
Ngram is 5
$ handbreadth sparkles basin wor results disarrange dried-up ripening repaired flash.
$ gytrash-like beings presents renamed emir quickly considers procrastinated perched high-backed.
$ impetuosity bored guise waned finely adorned 2001 blissful development appropriate.
$ notes whiskers don't beds complaint 'this shutting conveyance bonnet-strings cheese-cakes.
$ untimely remonstrance 1260.txt barmecide majestically march-spirit reality high-hung cavalier cadet.

Ngram is 4
$ neighbourhood midst for battle lord finally argument serious sadness luxuriant.
$ bargaining match pewter puzzles mountains sky trap-door dance pronounced negatived.
$ scowling brand readable recommended wailing indirectly dank movements newly-risen lulling.
$ horseman dished spontaneously works she's negligently half-phrases wistful orthodox freaks.
$ unavailing 'surely master-wave lock contumacy planted entreating unfurrowed les aspect.

Ngram is 3
$ hanging wife gale flexibility current cavillers assure chimney-piece ruffian miller.
$ illumination soothing bid accept numbers merits bonnets insolvable indulgent fought.
$ anticipate buckles sobs disappointment journeyings continued plague-cursed increases creatures sacred.
$ adhere half-expecting attending wed waiting-woman really skeletons visitation briefly faltering.
$ society's hesitatingly disposition unlucky apostrophising injuries apt swollen trained xix.

Ngram is 2
$ mr. shrewd observations co-operate legally yew disproportionate showed pledge drawer.
$ excellences heritage stubble directness 'to scream reconcilement has strove torn.
$ deigned click-click attempted supplying hummed feel abigail inflict distinguish constitute.
$ whey-faced chiselled bombazeen page lightning suffering renews persevere obvious omnipotent.
$ for roasting whitewashed _his_ headlong crying verses blindness motto heathen.

Ngram is 1
$ conceive cheese-cake aims north-of-england deception hearer baptized risked cowardly clergy.
$ _elder_ fellow-missionary showing disorder oppresses frieze hint extraordinary volcano-tops attribute.
$ rejoin scrag listener unsatisfied initials brilliant 5th sphynx foaming will.
$ incarnate piano trying cathedral ghostly cause emptying tea outward chiffone_.
$ cosy prominences speak brightness dictionary dragged dreamed exclusion wolfish sinner.

Note on the result: Initially, when I tried to remove punctuation from the original corpus, the results contained words that are combined with one another. I suspect that this may be caused by certain removal methods in nltk.

Overall conclusion: Given all four questions were done with the assumption of Laplace smoothing, the expectation from the Programming Question 1,2,3 were that the probability of the tokens in the corpus chosen will be lowered and the probability of the tokens in the corpus but will increase. The expectation for 4 is that there will exists significant differences between different ngram training. However, The results above shown that there is nothing significant difference between any of the n-grams above. The reason for this is probablity we took too much from the rich i.e. the Laplace Smoothing added too much to the denominator causing the probability to choose the subsequent word in the corpus to decrease drastically.