CSE 156 NLP Assignment1 Report

Name: Shiang Hu

PID: A53267858

1. Code:

2.Language Model Implementation:
Class Trigram implements the trigram model, the word generated next should have the conditional probability P(Wi | Wi-2, Wi-1) meaning the probability of words sequence Wi-2, Wi-1,Wi under the condition that Wi-2, Wi-1 is the sequence of words before Wi, so the trigram model considers the context of previous two words as appose to unigram which considers only the probability of the current word. To calculate the conditional probability, we created dictionaries with keys of three words tuple and dictionary of two words tuple which stores the appearance times of the sequence when we scan over the training corpus. Then the conditional probability can be calculated using the count of Wi-2, Wi-1, Wi divided by the count of Wi-2, Wi-1. For the words at the start of the sentence we append two starting string before the sentence [‘*’, ’*’]. This way for the starting word of the sentence we can keep count of the sequence [‘*’, ’*’, Wi] and [‘*’, ’*’]. And for the second word we keep count of the sequence [‘*’, Wi-1, Wi] and [‘*’, Wi-1]. This is how the model calculate the conditional probability for the first and second word.
At training, the trigram model reads in the training corpus as input and filters out words that appears below a certain threshold which can be tuned by the hyper-parameter “threshold”. If “threshold” is set to five, then at training the model will filter out words with appearance less than five times and the count of that word will be added to the count of a specially defined word: “UNK”. By filtering out these words, we ensure that our language model doesn’t generate these words at test time. We tune “threshold” by looking at the perplexity of the dev data set and the generated sample sentences.
For smoothing the first implementation is the Laplace Smoothing. After calculating the conditional probability, we add Laplace Smoothing to flatten the distribution of the words’ probability by the formula:

$$\frac{Count(W_{i-2}, W_{i-1}, W_i) + 1}{Count(W_{i-2}, W_{i-1}) + delta * |v|}$$

which gives words with less count a bit more probability and reduce the probability of frequently seen words. $|v|$ is the number of vocabulary in our training corpus. And delta is a hyper-parameter we tune so we don’t over-smooth the probabilities by distributing too much to less seen words. We tune both these parameters by looking at the perplexity produce by the in-domain text dev data set. In addition, the model returned the logarithmic probability like the Unigram, and for words sequence not seen in training data, the model returns log(1/(delta*|v|)). The second implementation is Linear Interpolation where the probability of the word is the combination of trigram, bigram and unigram i.e:
P(Wi | Wi-2, Wi-1) = lambda1*P(Wi | Wi-2, Wi-1) + lambda2*P(Wi | Wi-1) + lambda3*P(Wi)
for the lambda parameters we use grid search and find the best lambda combination that has the lowest perplexity among in-domain text dev data set for the in-domain text. This model also returns the logarithmic probability

3. Analysis of in-domain text:

Unigram train on brown corpus results of in-domain text:

Test data	brown train	brown dev	brown test
Perplexity	1513.8	1589.3	1604.2

Unigram train on reuters corpus results of in-domain text:

Test data	reuters train	reuters dev	reuters test
Perplexity	1471.2	1479.0	1500.6

Unigram train on gutenberg corpus results of out-of-domain text:

Test data	gutenberg train	gutenberg dev	gutenberg test
Perplexity	982.5	991.5	1005.7

Unigram train on brown corpus results of in-domain text:
sample 2: They with as be the Knight the New house

Trigram train Laplace Smoothing on brown corpus results of in-domain text, with fixed threshold = 4, delta=0.1:

Test data	brown train	brown dev	brown test
Perplexity	16.6	52.0	52.5

Trigram train Laplace Smoothing on reuters corpus results of in-domain text, with fixed threshold = 4, delta=0.1:

Test data	reuters train	reuters dev	reuters test
Perplexity	13.7	26.4	27.0

Trigram train Laplace Smoothing on gutenberg corpus results of in-domain text, with fixed threshold = 4, delta=0.1:

Test data	gutenberg train	gutenberg dev	gutenberg test
Perplexity	9.5	15.8	16.1

sample2: They Mount Protestantism watched sister masters clown arc Holmes Jenkins Coming vulgar cheerful subsistence useless Calvin Aid pursuit dominate archaic looks professionals sets butter drunk provision Guest analyze Burnside Nations evening forceful larvae extract difficulty Column Dean Price quack release begin amid boldly Manu auditorium predispositions inhabited doubts cheer dreams Revolution prevent mountains invariably urbanization comply Roberts carefree resting historian boating furnishings comic radically apportioned bears bases keel qualify wedge stick cop pick appreciation Fox compressed fingers Semitism two aside doubt sampled Buckley bitterness participates Recognizing Experimental unlikely absurd Liberal speck lied ensemble channels jealousy desperate margins collaborators during responded Hough North

---

Trigram train Linear Interpolation on brown corpus results of in-domain text, with fixed threshold = 4, lamda1= 0.015, lambda2=0.285, lambda3=0.7

Test data	brown train	brown dev	brown test
Perplexity	16.6	52.0	52.5

Trigram train Linear Interpolation on reuters corpus results of in-domain text, with fixed threshold = 4, lamda1= 0.015, lambda2=0.285, lambda3=0.7

Test data	reuters train	reuters dev	reuters test
Perplexity	13.7	26.4	27.0

Trigram train Linear Interpolation on gutenberg corpus results of in-domain text, with fixed threshold = 4, lamda1= 0.015, lambda2=0.285, lambda3=0.7

Test data	gutenberg train	gutenberg dev	gutenberg test
Perplexity	9.5	15.8	16.1

Trigram train Linear Interpolation on brown corpus:
sample 2: They understand being dispersed is their and situation the steps adjustment her sensors his one of UNK to sequences authorized simply we foreign let are called Department allows general two UNK book saying from accompanied It north of but has * cover By mate The as truth * month and Why was variety line the Washington It to be though be installed

Observation:
We observe that the Trigram Laplace Smoothing methods has the lowest perplexity of the three models and Trigram Linear interpolation come next then last is the Unigram. For the sample sentences, we observe that the Unigram has the shortest sentence length then comes Trigram Linear Interpolation then longest is the Trigram Laplace Smoothing. The cause of this is because Unigram only consider the frequency of a word and ‘End-Of-Sentence’ appears in every sentence which will be most likely to be chosen. As for Linear Interpolation the lambda values favors Unigram more and will produce bit longer length sentences than Unigram. For the Unigram it doesn’t consider context so its produced sentence seems to have the least sense, whereas Linear Interpolation will have the most sense since it considers context as in the Laplace Smoothing model but also produce sentence length that makes more sense.

Trigram train Laplace Smoothing on brown corpus results of in-domain text comparing different threshold, with fixed delta = 0.1:

threshold	Test data	brown train	brown dev	brown test
0	Perplexity	1543.3	2669.84	2674.47
threshold	Test data	brown train	brown dev	brown test
1	Perplexity	1543.3	2669.84	2674.47
threshold	Test data	brown train	brown dev	brown test
2	Perplexity	882.5	1587.0	1589.7
threshold	Test data	brown train	brown dev	brown test
3	Perplexity	636.6	1189.4	1191.3
threshold	Test data	brown train	brown dev	brown test
4	Perplexity	501.2	969.4	971.3
threshold	Test data	brown train	brown dev	brown test
5	Perplexity	414.8	825.8	827.5

Observation:
For larger threshold we observe that the perplexity gets lower, but the sample sentences make less sense. Since when the threshold gets bigger the model stores more “UNK” keys, and will produce less sense sentences. But the perplexity will get lower since producing “UNK” corresponds to the training model.

Trigram train Laplace Smoothing on brown corpus results of in-domain text comparing different deltas, with fixed threshold = 4:

Delta	Test data	brown train	brown dev	brown test
0.0001	Perplexity	6.1	2.6	2.6
Delta	Test data	brown train	brown dev	brown test
0.001	Perplexity	15.9	16.6	16.7
threshold	Test data	brown train	brown dev	brown test
0.01	Perplexity	75.0	119.7	120.0
threshold	Test data	brown train	brown dev	brown test
0.1	Perplexity	501.2	969.4	971.3
threshold	Test data	brown train	brown dev	brown test
0.3	Perplexity	1339.3	2709.4	2713.7
threshold	Test data	brown train	brown dev	brown test
0.6	Perplexity	2537.6	5225.1	5232.4
threshold	Test data	brown train	brown dev	brown test
1	Perplexity	4096.8	8513.2	8524.1

Observation:
We can see that when delta declines the perplexity declines to a point where training perplexity becomes larger than dev and test data.

4. Analysis of out-of-domain text:

Unigram train on brown corpus results of out-of-domain text:

Test data	reuters train	gutenberg train
Perplexity	6780.8	1758.0
Test data	reuters dev	gutenberg dev
Perplexity	6675.6	1739.4
Test data	reuters test	gutenberg test
Perplexity	6736.6	1762.0

Unigram train on reuters corpus results of out-of-domain text:

Test data	brown train	gutenberg train
Perplexity	3806.3	4882.8
Test data	brown dev	gutenberg dev
Perplexity	3808.8	4833.8
Test data	brown test	gutenberg test
Perplexity	3865.1	4887.4

Unigram train on gutenberg corpus results of out-of-domain text:

Test data	brown train	reuters train
Perplexity	2616.5	12420.1
Test data	brown dev	reuters dev
Perplexity	2604.2	12256.3
Test data	brown test	reuters test
Perplexity	2626.05	12392.5

Trigram train Linear Interpolation on brown corpus results of out-of-domain text, with fixed threshold = 4, delta=0.1:

Test data	reuters train	gutenberg train
Perplexity	1111.8	1049.6
Test data	reuters dev	gutenberg dev
Perplexity	1180.5	1118.3
Test data	reuters test	gutenberg test
Perplexity	1180.7	1115.1

Trigram train Linear Interpolation on reuters corpus results of out-of-domain text, with fixed threshold = 4, delta=0.1:

Test data	brown train	gutenberg train
Perplexity	970.4	1183.9
Test data	brown dev	gutenberg dev
Perplexity	1095.6	1237.1
Test data	brown test	gutenberg test
Perplexity	1098.5	1236.0

Trigram train Linear Interpolation on gutenberg corpus results of out-of-domain text, with

fixed threshold = 4, delta=0.1:

Test data	brown train	reuters train
Perplexity	987.6	1334.9
Test data	brown dev	reuters dev
Perplexity	1144.4	1407.7
Test data	brown test	reuters test
Perplexity	1145.7	1407.0

Trigram train Linear Interpolation on brown corpus results of out-of-domain text, with fixed threshold = 4, lamda1= 0.015, lambda2=0.285, lambda3=0.7:

Test data	reuters train	gutenberg train
Perplexity	262.1	48.5
Test data	reuters dev	gutenberg dev
Perplexity	404.1	68.9
Test data	reuters test	gutenberg test
Perplexity	410.4	69.9

Trigram train Linear Interpolation on reuters corpus results of out-of-domain text, with fixed threshold = 4, lamda1= 0.015, lambda2=0.285, lambda3=0.7:

Test data	brown train	gutenberg train
Perplexity	66.2	155.4
Test data	brown dev	gutenberg dev
Perplexity	143.6	220.2
Test data	brown test	gutenberg test
Perplexity	145.8	224.2

Trigram train Linear Interpolation on gutenberg corpus results of out-of-domain text, with fixed threshold = 4, lamda1= 0.015, lambda2=0.285, lambda3=0.7:

Test data	brown train	reuters train
Perplexity	32.2	429.1
Test data	brown dev	reuters dev
Perplexity	67.8	660.5
Test data	brown test	reuters test
Perplexity	68.6	674.5

Observation:

We observe that the perplexity Trigram Linear Interpolation > Trigram Laplace Smoothing > Unigram. Although at in-domain text Laplace Smoothing has the lowest perplexity for out-domain Linear Interpolation outperforms it. Apparently, Linear Interpolation is more flexible, or it is less prone to over fit so it performs better at testing.