CSE-143 Assignment 2

Harish Dukkipati
Hanlin Huang
Vivan Singhal
Zhengheng Li

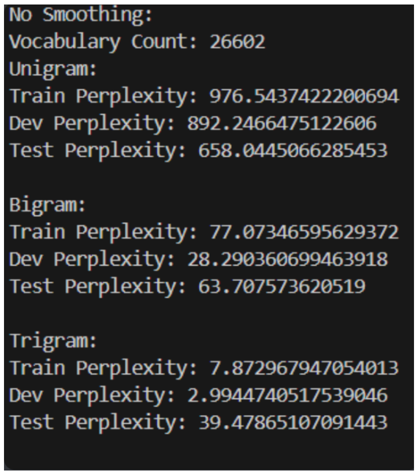
May 2024

1 Part 1: n-gram modeling

In our training, we scored 658.04 on the unigram perplexity, 63.71 on the bigram perplexity, and 39.48 on the trigram perplexity. This matches up with the expected values from the PDF, supporting that our models are correct. We also got the expected 26602 vocabulary count from the input, showing that we parsed our data correctly to achieve our results.

For our models, we used a function to scrape the data from the files initially, and then created 2 functions to get the data for unigrams and the bigrams and trigrams. We created two functions per n-gram for the MLE estimator and perplexity scores as well, bringing the total amount of functions used to 9.

We tested our momdels with the files provided and with the HDTV . line as suggested by the Assignment 2 PDF. Here is an example of our output.

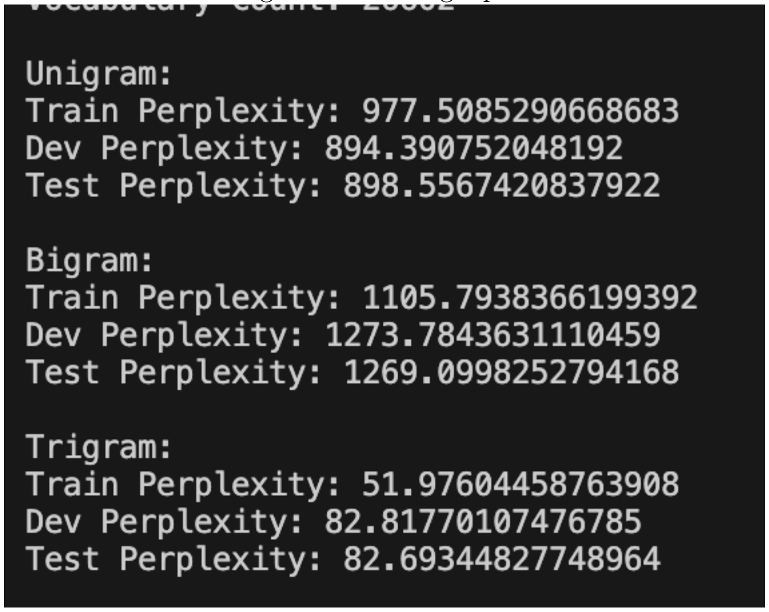


```
No Smoothing:
Vocabulary Count: 26602
Unigram:
Train Perplexity: 976.5437422200694
Dev Perplexity: 892.2466475122606
Test Perplexity: 658.0445066285453

Bigram:
Train Perplexity: 77.07346595629372
Dev Perplexity: 28.290360699463918
Test Perplexity: 63.707573620519

Trigram:
Train Perplexity: 7.872967947054013
Dev Perplexity: 2.9944740517539046
Test Perplexity: 39.47865107091443
```

2 Part 2: additive smoothing

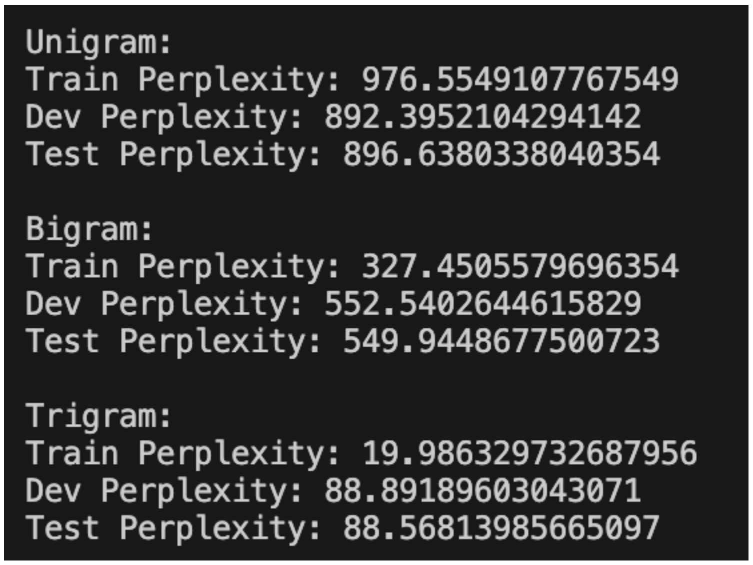
In order to add additive soothing to the unigram, bigram, and trigram models I used the additive soothing formula P(wn — wn-1) = (C(Wn-1, Wn) + alpha) / (C(Wn-1) + alpha * V), where C(Wn-1, Wn) is the count of the n-gram, V is the vocabulary size, and alpha is the soothing parameter. I used the same base functions that I already had from problem 1 and then added this technique for each ngram in my ngram model function.

Here are the results I got when setting alpha to 1



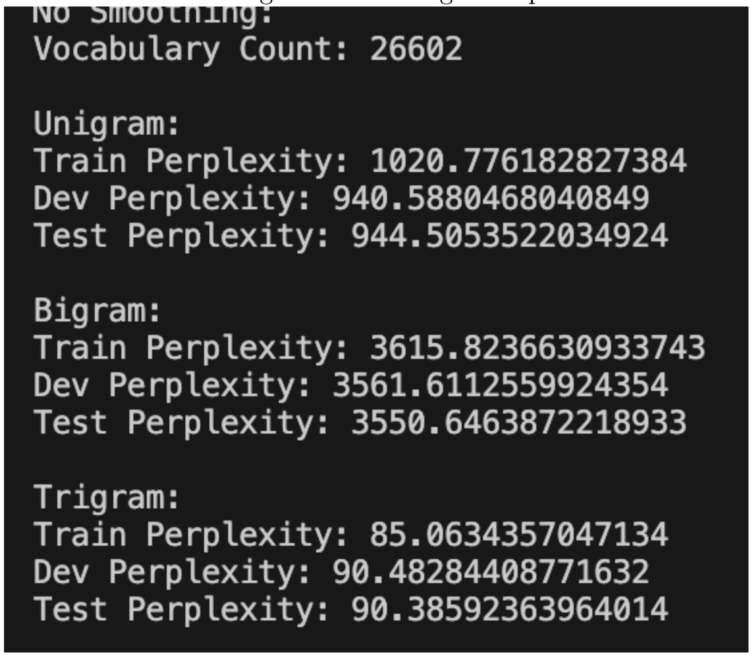
```
Vocabulary Count: 26602

Unigram:
Train Perplexity: 977.5085290668683
Dev Perplexity: 894.390752048192
Test Perplexity: 898.5567420837922

Bigram:
Train Perplexity: 1105.7938366199392
Dev Perplexity: 1273.7843631110459
Test Perplexity: 1269.0998252794168

Trigram:
Train Perplexity: 51.97604458763908
Dev Perplexity: 82.81770107476785
Test Perplexity: 82.69344827748964
```

Here are the results I got when setting alpha to 0.1



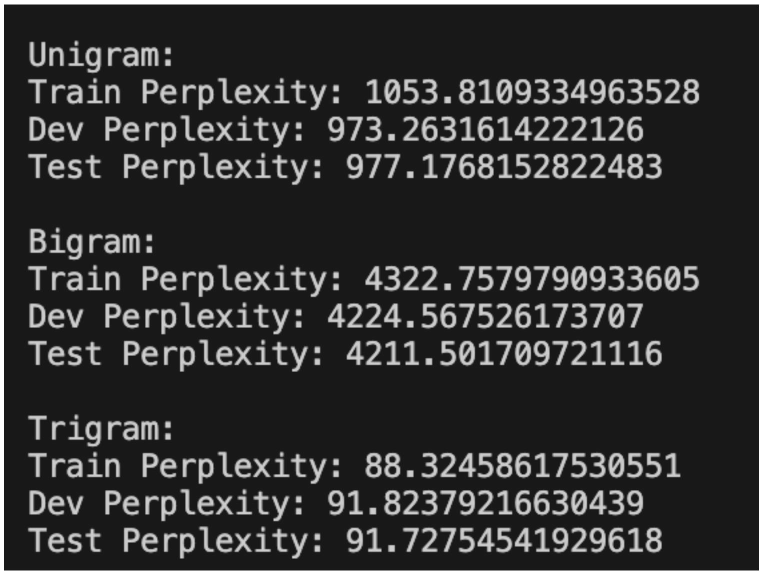
```
Unigram:
Train Perplexity: 976.5549107767549
Dev Perplexity: 892.3952104294142
Test Perplexity: 896.6380338040354

Bigram:
Train Perplexity: 327.4505579696354
Dev Perplexity: 552.5402644615829
Test Perplexity: 549.9448677500723

Trigram:
Train Perplexity: 19.986329732687956
Dev Perplexity: 88.89189603043071
Test Perplexity: 88.56813985665097
```

Here are the results I got when setting the alpha to 10



```
No Smoothing:
Vocabulary Count: 26602

Unigram:
Train Perplexity: 1020.776182827384
Dev Perplexity: 940.5880468040849
Test Perplexity: 944.5053522034924

Bigram:
Train Perplexity: 3615.8236630933743
Dev Perplexity: 3561.6112559924354
Test Perplexity: 3550.6463872218933

Trigram:
Train Perplexity: 85.0634357047134
Dev Perplexity: 90.48284408771632
Test Perplexity: 90.38592363964014
```

The best perplexity I got an my dev set is when I set my alpha to a higher number. One value in particular that gave me a better perplexity than the ones I tested before was setting the alpha to 15.



```
Unigram:
Train Perplexity: 1053.8109334963528
Dev Perplexity: 973.2631614222126
Test Perplexity: 977.1768152822483

Bigram:
Train Perplexity: 4322.7579790933605
Dev Perplexity: 4224.567526173707
Test Perplexity: 4211.501709721116

Trigram:
Train Perplexity: 88.32458617530551
Dev Perplexity: 91.82379216630439
Test Perplexity: 91.72754541929618
```

3 Part 3: linear interpolation