University of Pittsburgh

School of Computing and Information

CS2710

Natural Language Processing

Language Modeling

Date: October 28, 2023

Jacob Hoffman

Advisors:

Michael Yoder, PhD

Part 1:

In your report, include:

1. a description of how you wrote your program, including all assumptions and design decisions

For homework 3, I created a python class called NGrams(n, file, token_type), which I used to create a unigram, bigram, and trigram class instance. At first, I had worked on the project with the entire words as tokens instead of characters, but this was an accident and so I evolved the NGrams class to handle both possible pre-processing scenarios, i.e. the class can set up character or word token types. I used full word tokens to help me run the program faster at times since there were less tokens to process. Apparently, my project could be optimized to run in a faster time complexity. During pre-processing, I appended the ‘<s>’ and ‘</s>’ tokens to each line of the file. I chose to lower-case all characters, but besides that, I didn’t do any additional pre-processing. Inside the main.py file, there are functions declared to handle the file saving, the perplexity calculations, and the interpolated/raw unigram, bigram, and trigram probabilities. I feel that the way I set up these calculations could be optimized because they are kind of slow, especially whenever using the character token type. The language models that are saved have prefixes as rows, and postfixes (the current word) as columns.

2. an excerpt of the two untuned trigram language models for English, displaying all n-grams and their probability with the two-character history t h

En_trigrams.csv:

```
, ,!,n,],ó,t,;,[,á,>,l,7,x,q,h,6,5,?,(,a,1,",",m,-,u,:,',í,ä,),°,i,0,z,<,p,%,3,4,é,b,8,
"""",f,9,s,e,o,/,r,c,j,y,.,w,2,v,d,g,k


th,0.05411764705882353,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0005228758169934641,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.1257516339869281,0.0,0.001568627450980392,0.0,0.0,0.00392
156862745098,0.0,0.0,0.0,0.0,0.0,0.0,0.12156862745098039,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.00261437908496732,0.6603921568627451,0.018300653594771243,0.0
,0.005751633986928104,0.00026143790849673205,0.0,0.0018300653594771241,0.0018300653594
771241,0.0005228758169934641,0.0,0.0,0.0010457516339869282,0.0,0.0
```

En_interpolated_trigrams.csv:

```
,8,<,o,v,e,7,b,y,z,ó,n,],w,j,1,m,c,d,>,',u,:,"""",%,!,(,á,í,x,a,t,3,s,g,?,
,p,4,;,l,2,h,f,i,-,9,0,5,é,ä,r,.,q,k,6,°,",",/,[,)


th,6.266348093615198e-05,0.004042805221687224,0.038036790259539634,0.00270665809591959
63,0.31130414862810407,4.042805221687224e-05,0.003222115761684717 3,0.0102164561270112 0
2,8.894171487711893e-05,1.010701305421806e-05,0.019205450670876364,1.4149818275905284e
-05,0.004550430494018378 6,0.002083739400448718 7,0.00027895356029641844,0.0081907025839
12302,0.008356131070164509,0.009146206527345938,0.004042805221687224,0.00341535407665 5
9405,0.010843852620037 69,3.840664960602863e-05,2.4256831330123345e-05,2.82996365518105
7e-05,8.085610443374448e-06,7.074909137952642e-05,6.064207832530836e-06,2.021402610843
612e-06,0.000521521873597652,0.0985532232468617 2,0.027003955767931 23,3.840664960602863
e-05,0.022200625725083162,0.004170153586170372,4.851366266024669e-05,0.079747222475251
65,0.006775741551547788,4.851366266024669e-05,3.43638443843414e-05,0.0107800256336793 6
1,0.00018394763758676 87,0.01192241924274456 4,0.005421401802282568,0.08769844631921285,
0.0019502528811087833,0.000337574236010883 2,0.000293103378572323 73,9.50059227096497 6e-
05,4.042805221687224e-06,0.200010107013054 2,0.020207441146606716,0.006683453434628998,
0.000307253196848229,0.0012694408396097883,5.862067571446475e-05,6.064207832530836e-06
,0.010394258953489024,0.002090130299612295,1.4149818275905284e-05,8.085610443374448e-0
5
```

Bigram ‘th’ history would be the distribution for the ‘h’ row

En_bigrams.csv:

```
,
,!,n,],ó,t,;,[,á,>,1,7,x,q,h,6,5,?,(,a,1,",",m,-,u,:,',í,ä,),°,i,0,z,<,p,%,3,4,é,b,8,"
""",f,9,s,e,o,/,r,c,j,y,.,w,2,v,d,g,k


h,0.11601362862010221,0.0,0.0011925042589437820,0.0,0.0,0.651618398637138,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.00017035775127768313,0.0,0.0,0.0,0.0,0.00034071550255536625,0.0,0
.0,0.0,0.00017035775127768313,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.00017035775127768313,0.0,0
.0,0.0,0.004599659284497446,0.0,0.0,0.0,0.0,0.0,0.0,0.00017035775127768313,0.00017035
775127768313,0.0,0.04224872231686542,0.004599659284497446,0.0076660988074957410,0.0,0.
00051107325383304940,0.07512776831345826,0.0,0.0,0.0,0.07325383304940375,0.0,0.0,0.0003
4071550255536625,0.021465076660988076,0.00017035775127768313
```

En_interpolated_bigrams.csv

```
,8,<,o,v,e,7,b,y,z,ó,n,],w,j,1,m,c,d,>,',u,:,"""",%,!,(,á,í,x,a,t,3,s,g,?,
,p,4,;,l,2,h,f,i,-,9,0,5,é,ä,r,.,q,k,6,°,",",/,[,)

h,0.017798449988478005,0.017798449988478005,0.021631499392225875,0.017798449988478005,0.020098279630726728,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.018394702117949895,0.017798449988478005,0.054425366513179876,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.05536233414520714,0.017968807739755687,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017883628864116846,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017968807739755687,0.343607649307047,0.017798449988478005,0.038922811146910714,0.028530988318972043,0.017798449988478005,0.07580526429852912,0.020098279630726728,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017883628864116846,0.017883628864116846,0.017883628864116846,0.017883628864116846,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.01805398661539453,0.017798449988478005,0.017798449988478005,0.017883628864116846,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017798449988478005,0.017798449988478005
```

3. documentation that your probability distributions are valid (sum to 1)

Please run the program to verify the output, but here are the terminal outputs that sum the probability distributions for each model:

Interpolated (please note that the trigrams interpolation are summing slightly above 1.0):

```
(nlp-env) jacobhoffman@Jacobs-MBP-2 cs2731 % python3 hw3/main.py --debug --save --interpolation
current file path:  datasets/hw3_data/training.en
current file path:  datasets/hw3_data/training.es
current file path:  datasets/hw3_data/training.de
current file path:  datasets/hw3_data/test

# training.en file
unigrams total probability:  0.9999999999999999
bigrams interpolated total probabilities summed / vocab_count:  0.9999958333333333
trigrams interpolated total probabilities summed / bigram_words_count:  1.27327370543255
saving en interpolated probabilities models to file

# training.es file
unigrams total probability:  1.0
bigrams interpolated total probabilities summed / vocab_count:  0.99999609375
trigrams interpolated total probabilities summed / bigram_words_count:  1.3256368567916608
saving es interpolated probabilities models to file

# training.de file
unigrams total probability:  1.0
bigrams interpolated total probabilities summed / vocab_count:  0.9999959016393443
trigrams interpolated total probabilities summed / bigram_words_count:  1.2720555203782924
saving de interpolated probabilities models to file
```

Uninterpolated:

```
(nlp-env) jacobhoffman@Jacobs-MBP-2 cs2731 % python3 hw3/main.py --debug --save
current file path:  datasets/hw3_data/training.en
current file path:  datasets/hw3_data/training.es
current file path:  datasets/hw3_data/training.de
current file path:  datasets/hw3_data/test

# training.en file
unigrams total probability:  1.0
bigrams total probabilities summed / vocab_count:  0.9999916666666666
trigrams total probabilities summed / bigram_words_count:  0.999999391727494
saving en  probabilities models to file

# training.es file
unigrams total probability:  1.0
bigrams total probabilities summed / vocab_count:  0.9999921875
trigrams total probabilities summed / bigram_words_count:  0.9999994292237443
saving es  probabilities models to file

# training.de file
unigrams total probability:  1.0000000000000002
bigrams total probabilities summed / vocab_count:  0.9999918032786885
trigrams total probabilities summed / bigram_words_count:  0.9999994646680941
saving de  probabilities models to file
```

4. for all your unsmoothed and smoothed models, the average perplexity score across all lines in the test document.

For the perplexity function in my program, I completely skipped over any prefixes or suffixes (characters/n-grams) that were not present in the model vocabulary. Additionally, any probabilities that were selected but equaled zero were skipped. I feel that the perplexity scores are valid, but the scores for the mismatched language models seem too low. I am unsure if this is true, though.

Please run the program to verify the perplexity output, but here are the terminal outputs that sum the probability distributions for each model:

Interpolated:

- Although the interpolated trigram model summed above 1, it seems to be valid when used to calculate the perplexity.

```
en  unigram model average perplexity:  0.17777359357923317
en  bigram model average perplexity:  0.16936451576065925
en  trigram model average perplexity:  0.06720471862378073

es  unigram model average perplexity:  0.19994968588992088
es  bigram model average perplexity:  0.1673879655672906
es  trigram model average perplexity:  0.15015435213046283

de  unigram model average perplexity:  0.17925463007446493
de  bigram model average perplexity:  0.15628525501468343
de  trigram model average perplexity:  0.12493045506622076
```

Uninterpolated:

```
en  unigram model average perplexity:  0.17777359357923317
en  bigram model average perplexity:  0.19106580922403132
en  trigram model average perplexity:  0.052077320358346404

es  unigram model average perplexity:  0.19994968588992088
es  bigram model average perplexity:  0.35468818068008434
es  trigram model average perplexity:  0.07121355646266139

de  unigram model average perplexity:  0.17925463007446493
de  bigram model average perplexity:  0.17842763945919748
de  trigram model average perplexity:  0.08573738486762075
```

5. generated text outputs for the following inputs: bigrams starting with 10 letters of your choice, and trigrams using those 10 letters as the first character with a second meaningful character of your choice.

Part 2:

In your report, include:

1. critical analysis of your language identification results: e.g., why do your perplexity scores tell you what language the test data is written in? what does a comparison of your unsmoothed versus smoothed scores tell you about which performs best? what does a comparison of your unigram, bigram, and trigram scores tell you about which performs best? Etc.

The perplexity scores allow us to determine the language that the test data is written in because it is a calculation of how predictable the text is based on the language models. I feel that the interpolated models are a bit more accurate because their accuracy is not as volatile (unigram->bigram->trigram barely changes in the mismatched, interpolated language models vs the uninterpolated, which fluctuate more). Finally, the unigram models are practically useless for determining the test file's language, but this kind of makes sense because the three language models have a very similar vocabulary (especially when considering single characters, i.e. unigram vocabulary).

2. critical analysis of your generation results: e.g., are there any difference between the sentences generated by bigrams and trigrams, or by the unsmoothed versus smoothed models? Give examples to back up your conclusions.