Natural Language Processing - Home Work 2

Name: Mani Deep Cherukuri
StudentId: 5829680
Email: cheru050@gmail.com

Introduction:
In this report, we try to build a n-gram based language model by writing a program that will classify authors based on a training text.

Encoding type:
For the assignment, I used the utf-8 encoding**.**

Information of the language model:
The models have been trained on both bigram and trigram, using 90% of the text data from the authors which is trained on one model at a time. The remaining 10% was used as a development set for assessment. We have four authors, thus we have four of these fundamental MLE-based models. I have tried 4 additional forms of smoothing on each author, thus there are a total of 5 models including the base model MLE for each author. Therefore, we have 20 trained models for each gram in total data. If you include both bigram and trigram, it is 40 for all 4 authors.

Smoothing Technique:

I have used 4 different methods of smoothing exclude the base MLE to decided the best method based on their classification accuracies. The smoothing methods used are:
1. Laplace Smoothing
2. Stupid Backoff
3. Witten Bell Interpolation
4. Lidstone Smoothing

Default values set in the code are Lidstone for bigram model.

Dealing with the OOV words during runtime:

The MLE model returns an extremely high or infinite perplexity score for words that are not in the vocabulary. I used smoothing techniques, which gives them a weight greater than zero, to deal with this. For eg, Lidstone smoothing which is designed to handle the problem of zero probabilities for n-grams that appear in the training data helps in dealing with OOV words during runtime in language modelling.

Tweaks to improve results:
Sometimes, results on the testing accuracy may improve when we encoded the input text as bigrams instead of every grams produce better results.

How to RUN: Use the commands displayed on the top line of every respective images to generate the results. Also, change the n values and smoothing name inside the code if you want to run different smoothing models or trigram.

Accuracy results without the flag:

N = 2

```
(base) lucy@Lucys-MacBook-Air-74 Final % python3 classifier.py authorlist
splitting into training and development datasets
training LMs...(this may take a while)
ngram is : 2, model is : LIDSTONE
-----------------Accuracy part without test flag--------------------
austen_utf8     90.6% correct
dickens_utf8    71.0% correct
tolstoy_utf8    78.9% correct
wilde_utf8      66.3% correct
```

N = 3

```
(base) lucy@Lucys-MacBook-Air-74 Final % python3 classifier.py authorlist
splitting into training and development datasets
training LMs...(this may take a while)
ngram is : 3, model is : LIDSTONE
-----------------Accuracy part without test flag--------------------
austen_utf8     88.8% correct
dickens_utf8    57.9% correct
tolstoy_utf8    54.1% correct
wilde_utf8      66.5% correct
```

Classification Results for the sample testfile.txt (mixed author sentences) and their perplexity scores:

N = 2

```
(base) lucy@Lucys-MacBook-Air-74 Final % python3 classifier.py authorlist -test testfile.txt
training LMs...(this may take a while)
ngram is : 2, model is : LIDSTONE
-------------Classification part with test flag and testfile----------------
Sentence is : ['i', 'know', 'he', 'would', 'be', 'hurt', 'by', 'my', 'failing', 'in', 'such', 'a', 'mark', 'of', 'respect', 'to', 'him', 'on', 'the', 'present', 'occas
ion']
Predicted author is : austen_utf8
perplexity for the above sentence :172.5988696693512
--------------------------------------
Sentence is : ['i', 'have', 'come', 'back', 'sir', 'as', 'you', 'anticipate', 'pursuing', 'the', 'object', 'that', 'took', 'me', 'away']
Predicted author is : dickens_utf8
perplexity for the above sentence :256.991119804979
--------------------------------------
Sentence is : ['to', 'be', 'able', 'to', 'crush', 'it', 'absolutely', 'he', 'awaited', 'the', 'arrival', 'of', 'the', 'rest', 'of', 'the', 'troops', 'who', 'were', 'on
', 'their', 'way', 'from', 'vienna', 'and', 'with', 'this', 'object', 'offered', 'a', 'three', 'days', 'truce', 'on', 'condition', 'that', 'both', 'armies', 'should',
'remain', 'in', 'position', 'without', 'moving']
Predicted author is : tolstoy_utf8
perplexity for the above sentence :398.22583628099784
--------------------------------------
Sentence is : ['as', 'he', 'thought', 'of', 'it', 'a', 'sharp', 'pang', 'of', 'pain', 'struck', 'through', 'him', 'like', 'a', 'knife', 'and', 'made', 'each', 'delicat
e', 'fibre', 'of', 'his', 'nature', 'quiver']
Predicted author is : wilde_utf8
perplexity for the above sentence :267.2104277988832
--------------------------------------
Sentence is : ['depend', 'upon', 'it', 'emma', 'a', 'sensible', 'man', 'would', 'find', 'no', 'difficulty', 'in', 'it']
Predicted author is : austen_utf8
perplexity for the above sentence :257.3892615502083
--------------------------------------
```

N = 3

```
(base) lucy@Lucys-MacBook-Air-74 Final % python3 classifier.py authorlist -test testfile.txt
training LMs...(this may take a while)
ngram is : 3, model is : LIDSTONE
-------------Classification part with test flag and testfile----------------
Sentence is : ['i', 'know', 'he', 'would', 'be', 'hurt', 'by', 'my', 'failing', 'in', 'such', 'a', 'mark', 'of', 'respect', 'to', 'him', 'on', 'the', 'present', 'occas
ion']
Predicted author is : austen_utf8
perplexity for the above sentence :341.38996084118185
--------------------------------------
Sentence is : ['i', 'have', 'come', 'back', 'sir', 'as', 'you', 'anticipate', 'pursuing', 'the', 'object', 'that', 'took', 'me', 'away']
Predicted author is : dickens_utf8
perplexity for the above sentence :381.6637276986109
--------------------------------------
Sentence is : ['to', 'be', 'able', 'to', 'crush', 'it', 'absolutely', 'he', 'awaited', 'the', 'arrival', 'of', 'the', 'rest', 'of', 'the', 'troops', 'who', 'were', 'on
', 'their', 'way', 'from', 'vienna', 'and', 'with', 'this', 'object', 'offered', 'a', 'three', 'days', 'truce', 'on', 'condition', 'that', 'both', 'armies', 'should',
'remain', 'in', 'position', 'without', 'moving']
Predicted author is : tolstoy_utf8
perplexity for the above sentence :715.374425601216
--------------------------------------
Sentence is : ['as', 'he', 'thought', 'of', 'it', 'a', 'sharp', 'pang', 'of', 'pain', 'struck', 'through', 'him', 'like', 'a', 'knife', 'and', 'made', 'each', 'delicat
e', 'fibre', 'of', 'his', 'nature', 'quiver']
Predicted author is : wilde_utf8
perplexity for the above sentence :545.1129962162928
--------------------------------------
Sentence is : ['depend', 'upon', 'it', 'emma', 'a', 'sensible', 'man', 'would', 'find', 'no', 'difficulty', 'in', 'it']
Predicted author is : austen_utf8
perplexity for the above sentence :373.4088534477695
--------------------------------------
```

Generation of 5 samples and their perplexity:
Observed that WBI is generating samples with low perplexity. Uncomment the code to run it.

```
(base) lucy@Lucys-MacBook-Air-74 Final % python3 classifier.py authorlist -test testfile.txt
training LMs...(this may take a while)
ngram is : 2, model is : WBI
---------------Generation part of 5 samples for each author-----------------
--- For sample prompt 1 generating---
 Author : austen_utf8.txt -> ['thing', 'that', 'is', 'extremely', 'glad', 'to', 'separate', 'miss', 'fairfax', 'on']
 Perplexity : 61.8
 Author : dickens_utf8.txt -> ['there', 'was', 'in', 'five', 'minutes', 'afterwards', 'then', 'ham', 'i', 'must']
 Perplexity : 56.0
 Author : tolstoy_utf8.txt -> ['then', 'that', 'interest', 'heror', 'by', 'lawyers', 'overhanging', 'bright', 'flowers', 'he']
 Perplexity : 54.9
 Author : wilde_utf8.txt -> ['them', 'simply', 'disgraceful', 'of', 'ordinary', 'psychologists', 'tell', 'me', 'if', 'she']
 Perplexity : 60.0
--- For sample prompt 2 generating---
 Author : austen_utf8.txt -> ['and', 'think', 'the', 'fair', 'performers', 'countenance', 'of', 'the', 'case', '</s>']
 Perplexity : 48.6
 Author : dickens_utf8.txt -> ['and', 'threatening', 'waters', 'and', 'most', 'he', 'said', 'sydney', 'carton', '</s>']
 Perplexity : 45.5
 Author : tolstoy_utf8.txt -> ['and', 'thought', 'rostv', 'bowed', 'his', 'heels', 'stood', 'there', 'and', 'again']
 Perplexity : 64.9
 Author : wilde_utf8.txt -> ['against', 'them', 'sometimes', 'harry', 'how', 'i', 'really', 'the', 'centre', 'of']
 Perplexity : 95.2
--- For sample prompt 3 generating---
 Author : austen_utf8.txt -> ['who', 'will', 'ask', 'above', 'respect', 'to', 'of', 'her', 'of', 'ones']
 Perplexity : 96.0
 Author : dickens_utf8.txt -> ['which', 'were', 'again', '</s>', 'the', 'said', 'of', 'him', 'most', 'of']
 Perplexity : 126.9
 Author : tolstoy_utf8.txt -> ['which', 'was', 'a', 'celebrated', 'traveler', 'in', 'the', 'forms', 'in', 'such']
 Perplexity : 79.7
 Author : wilde_utf8.txt -> ['whom', 'were', 'a', 'cigarette', 'i', 'should', 'like', 'breakages', 'answered', 'lord']
 Perplexity : 61.7
--- For sample prompt 4 generating---
 Author : austen_utf8.txt -> ['common', 'flirt', 'that', 'one', 'of', 'all', '</s>', 'there', 'he', 'entered']
 Perplexity : 133.6
 Author : dickens_utf8.txt -> ['coach', 'my', 'face', 'of', 'roads', '</s>', '</s>', 'the', 'engines', 'but']
 Perplexity : 99.8
 Author : tolstoy_utf8.txt -> ['called', 'me', 'for', 'so', 'on', 'a', 'bad', 'that', 'he', 'felt']
 Perplexity : 79.2
 Author : wilde_utf8.txt -> ['button', 'hole', 'in', 'smiling', 'i', 'am', '</s>', 'the', 'fine', 'and']
 Perplexity : 90.7
--- For sample prompt 5 generating---
 Author : austen_utf8.txt -> ['come', 'all', 'know', 'bequeathed', 'all', 'less', 'tranquil', 'tone', 'that', 'he']
 Perplexity : 68.5
 Author : dickens_utf8.txt -> ['clapping', 'a', 'key', 'in', 'a', 'kiss', 'my', 'possible', 'to', 'dinner']
 Perplexity : 129.4
 Author : tolstoy_utf8.txt -> ['by', 'an', 'evening', '</s>', '<s>', 'i', 'was', 'strange', 'surroundings', 'a']
 Perplexity : 68.4
 Author : wilde_utf8.txt -> ['but', 'dont', 'know', 'at', 'all', 'mr', 'w', 'h', 'the', 'eminent']
 Perplexity : 41.1
```

Extra credits:
1. Without NLTK lib:
I have attached the classifier_scratch.py which doesn't use the nltk library and can be run using the same commands as above.

Results and Analysis on other n-gram models/smoothing techniques:

LAPLACE:
N = 2

```
(base) lucy@Lucys-MacBook-Air-74 Final % python3 classifier.py authorlist
splitting into training and development datasets
training LMs...(this may take a while)
ngram is : 2, model is : LAPLACE
-----------------Accuracy part without test flag--------------------
austen_utf8      91.1% correct
dickens_utf8     58.3% correct
tolstoy_utf8     66.0% correct
wilde_utf8       60.1% correct
```

N = 3

```
(base) lucy@Lucys-MacBook-Air-74 Final % python3 classifier.py authorlist
splitting into training and development datasets
training LMs...(this may take a while)
ngram is : 3, model is : LAPLACE
-----------------Accuracy part without test flag--------------------
austen_utf8      94.5% correct
dickens_utf8     32.6% correct
tolstoy_utf8     24.3% correct
wilde_utf8       55.3% correct
```

Stupid Backoff (SB):

N = 2

```
(base) lucy@Lucys-MacBook-Air-74 Final % python3 classifier.py authorlist
splitting into training and development datasets
training LMs...(this may take a while)
ngram is : 2, model is : SB
-----------------Accuracy part without test flag--------------------
austen_utf8      86.9% correct
dickens_utf8     43.2% correct
tolstoy_utf8     55.7% correct
wilde_utf8       50.7% correct
```

N = 3

```
(base) lucy@Lucys-MacBook-Air-74 Final % python3 classifier.py authorlist
splitting into training and development datasets
training LMs...(this may take a while)
ngram is : 3, model is : SB
-----------------Accuracy part without test flag--------------------
austen_utf8      86.9% correct
dickens_utf8     42.6% correct
tolstoy_utf8     57.0% correct
wilde_utf8       50.7% correct
```

Witten Bell Interpolation(WBI):

N = 2

```
(base) lucy@Lucys-MacBook-Air-74 Final % python3 classifier.py authorlist
splitting into training and development datasets
training LMs...(this may take a while)
ngram is : 2, model is : WBI
-----------------Accuracy part without test flag--------------------
austen_utf8      88.3% correct
dickens_utf8     46.1% correct
tolstoy_utf8     56.2% correct
wilde_utf8       45.1% correct
```

N = 3

```
(base) lucy@Lucys-MacBook-Air-74 Final % python3 classifier.py authorlist
splitting into training and development datasets
training LMs...(this may take a while)
ngram is : 3, model is : WBI
-----------------Accuracy part without test flag--------------------
austen_utf8      87.6% correct
dickens_utf8     47.7% correct
tolstoy_utf8     57.8% correct
wilde_utf8       49.3% correct
```

Analysis: From the above accuracy values, it is observed that the Laplace model of trigram produces the best results.

References:

https://www.kaggle.com/code/alvations/n-gram-language-model-with-nltk
https://medium.com/mti-technology/n-gram-language-model-b7c2fc322799
https://eliteai.medium.com/building-n-gram-language-model-from-scratch-9a5ec206b520
ChatGPT