INLP Assignment-1
Language Modelling & Tokenization

MANGLESH PATIDAR 2023201059

Tokenizer.py

```
manglesh@patidar:~/Desktop/SEM2/NLP/Assignment/2023201059_assignment1$ python3 tokenizer.py
INIT
Enter a sentence Is that what you mean? I am unsure.
[['Is', 'that', 'what', 'you', 'mean', '?'], ['I', 'am', 'unsure', '.']]
Enter a sentence
```

In language_model.py

language_model.py <lm_type><corpus_path>

Using “Pride and Prejudice” corpus:

i for Interpolation Model.

```
manglesh@patidar:~/Desktop/SEM2/NLP/Assignment/2023201059_assignment1$ python3 language_model.py tr ./Pride.txt
language_model.py tr ./Pride.txt
INIT
Average Perplexity: 103.56024666173481
manglesh@patidar:~/Desktop/SEM2/NLP/Assignment/2023201059_assignment1$ python3 language_model.py i ./Pride.txt
language_model.py i ./Pride.txt
INIT
Enter a sentence (type 'exit' to quit): I am a woman.
probability with linear interpolation:  3.300034897711514e-08
Enter a sentence (type 'exit' to quit):
```

g for Good-Turing Smoothing Model

```
manglesh@patidar:~/Desktop/SEM2/NLP/Assignment/2023201059_assignment1$ python3 language_model.py g ./Pride.txt
language_model.py g ./Pride.txt
INIT
Enter a sentence (type 'exit' to quit): I am a woman.
probaility with good turing smoothing:  4.848017787296747e-06
Enter a sentence (type 'exit' to quit):
```

Using On “Ulysses James” corpus:

i for Interpolation Model.

```
manglesh@patidar:~/Desktop/SEM2/NLP/Assignment/2023201059_assignment1$ python3 language_model.py tr ./James.txt
language_model.py tr ./James.txt
INIT
Average Perplexity: 123.73480183199938
manglesh@patidar:~/Desktop/SEM2/NLP/Assignment/2023201059_assignment1$ python3 language_model.py i ./James.txt
language_model.py i ./James.txt
INIT
Enter a sentence (type 'exit' to quit): I am a woman.
probability with linear interpolation:  2.496072923716027e-08
Enter a sentence (type 'exit' to quit):
```

g for Good-Turing Smoothing Model

```
manglesh@patidar:~/Desktop/SEM2/NLP/Assignment/2023201059_assignment1$ python3 language_model.py g ./James.txt
language_model.py g ./James.txt
INIT
Enter a sentence (type 'exit' to quit): I am a woman.
probaility with good turing smoothing:  0.0001753753172571115
Enter a sentence (type 'exit' to quit):
```

In generator.py generator.py <lm_type> <corpus_path> <k>

k denotes the number of candidates for the next word to be print.

Using “Pride and Prejudice” corpus:

i for Interpolation Model.

```
manglesh@patidar:~/Desktop/SEM2/NLP/Assignment/2023201059_assignment1$ python3 generator.py i ./Pride.txt 3
generator.py i ./Pride.txt 3
INIT
Enter a sentence to generate or enter exit: How are you
[('are', 0.05416643517291009), ('must', 0.041907251785293725), ('talking', 0.039220859028757396)]
Enter a sentence to generate or enter exit: I am
[('sure', 0.17219911830242512), ('not', 0.1037739447633395), ('afraid', 0.04522036297541945)]
Enter a sentence to generate or enter exit: I like
[('her', 0.1859262586718498), ('him', 0.17756127116479573), ('them', 0.16285388746375165)]
Enter a sentence to generate or enter exit: what do you
[('think', 0.08598653350499462), ('mean', 0.08467276752420017), ('know', 0.0387856116953307)]
Enter a sentence to generate or enter exit:
```

g for Good-Turing Smoothing Model

```
manglesh@patidar:~/Desktop/SEM2/NLP/Assignment/2023201059_assignment1$ python3 generator.py g ./Pride.txt 3
generator.py g ./Pride.txt 3
INIT
Enter a sentence to generate or enter exit: How are
{'you': 1.0, 'colonels': 0.9017939042089985, 'fees': 0.9017939042089985}
Enter a sentence to generate or enter exit: I like
{'him': 0.3333333333333333, 'them': 0.3333333333333333, 'her': 0.3333333333333333}
Enter a sentence to generate or enter exit:
```

Using On “Ulysses James” corpus:

i for Interpolation Model.

```
manglesh@patidar:~/Desktop/SEM2/NLP/Assignment/2023201059_assignment1$ python3 generator.py i ./James.txt 4
generator.py i ./James.txt 4
INIT
Enter a sentence to generate or enter exit: How are
[('you', 0.228301382079737 5), ('things', 0.13553890886971479), ('the', 0.1171144503053722), ('all', 0.09584665168884994)]
Enter a sentence to generate or enter exit: what can i
[('its', 0.16327462845801344), ('help', 0.16316561490279483), ('raise', 0.16315231009881143), ('was', 0.02694990675550112)]
Enter a sentence to generate or enter exit: what can you
[('<EOS>', 0.08548382306189974), ('see', 0.0838900018590437), ('do', 0.04446223989425266), ('in', 0.04351309867349667)]
Enter a sentence to generate or enter exit:
```

g for Good-Turing Smoothing Model

```
manglesh@patidar:~/Desktop/SEM2/NLP/Assignment/2023201059_assignment1$ python3 generator.py g ./James.txt 4
generator.py g ./James.txt 4
INIT
Enter a sentence to generate or enter exit: How are
{'you': 0.4342404696642773, 'things': 0.2898604014192874, 'the': 0.13794956445821766, 'all': 0.13794956445821766}
Enter a sentence to generate or enter exit:
```

Report Perplexity Scores

1. **On “Pride and Prejudice” corpus:**
    1. LM 1: Tokenization + 3-gram LM + Good-Turing Smoothing
        1. Train– 4.300661406461312
        2. Test - 2.0904711622577774
    2. LM 2: Tokenization + 3-gram LM + Linear Interpolation
        1. Train – 6.670651959858675
        2. Test - 127.93983394745626
2. **On “Ulysses James” corpus:**
    1. i. LM 3: Tokenization + 3-gram LM + Good-Turing Smoothing
        1. Train - 5.021805077744325
        2. Test – 2.4248484817284806
    2. ii. LM 4: Tokenization + 3-gram LM + Linear Interpolation
        1. Train - 6.7261515713523155
        2. Test – 123.73480183199926

Observations

1. The perplexity of test for good Turing smoothing is lower than train as the probabilities assigned to the unseen n-grams is very high as we used the formula p0 = N1/N which assigned a very high probability to unseen trigrams. As the test sentences may not have been seen before while training the probability of test sentences comes out to be high which brings down the average perplexity perplexity .
2. The perplexities for train and test in case of linear interpolation smoothing look valid.

Generation examples:

test sentence – how are, k=10

predictions without smoothing:
('you', 0.21818371120541644), ('the', 0.10909185560270822), ('things', 0.10909185560270822), ('all', 0.05454592780135411), ('grub', 6.30511676014699e-16), ('charmed', 6.30511676014699e-16), ('bluest', 6.30511676014699e-16), ('neat', 6.30511676014699e-16), ('pretended', 6.30511676014699e-16), ('clarke', 6.30511676014699e-16)

predictions using linear interpolation:
('you', 0.2675603573839283), ('the', 0.13686307364337272), ('things', 0.1106105691257712 3), ('all', 0.060260753130291896), ('<EOS>', 0.02612358593414 65), ('<SOS>', 0.016924287635176036), ('not', 0.01510365988637 0156), ('a', 0.014549737136501142), ('in', 0.00801012842485 5014), ('they', 0.007785786516151883)

predictions using good turing :
{'the': 0.18717869841877266, 'you': 0.6009622802954314, 'all': 0.024680322867023187, 'things': 0.18717869841877266, 'arator': 0.9240811986989467, 'remerciez': 0.9240811986989467, 'extinction': 0.9240811986989467, 'links': 0.9240811986989467, 'waggons': 0.9240811986989467, 'crystallised':

0.9240811986989467} Observations

1. with no smoothing there are more rare combinations of the words as I have assigned a very low probability for unseen trigrams if it does not exists
2. Linear interpolation gives words whose bigrams and unigrams that are frequently seen throughtout the corpus.
3. using good turing the words which occured together with the given context have appeared along with random unkown words.