https://drive.google.com/drive/folders/1O4YowcEMxVuCeGta70n9BWGfVevNKuRL?usp=sharing

2023201058_LM1_test-perplexity.txt :
Average Test Perplexity: 1005.5690505171402

2023201058_LM1_test-perplexity.txt :
Average Train Perplexity: 125834.41406153486

2023201058_LM2_test-perplexity.txt :
Average Test Perplexity: 5556.928637051639

2023201058_LM2_train-perplexity.txt :
Average Train Perplexity: 6.653584323764017

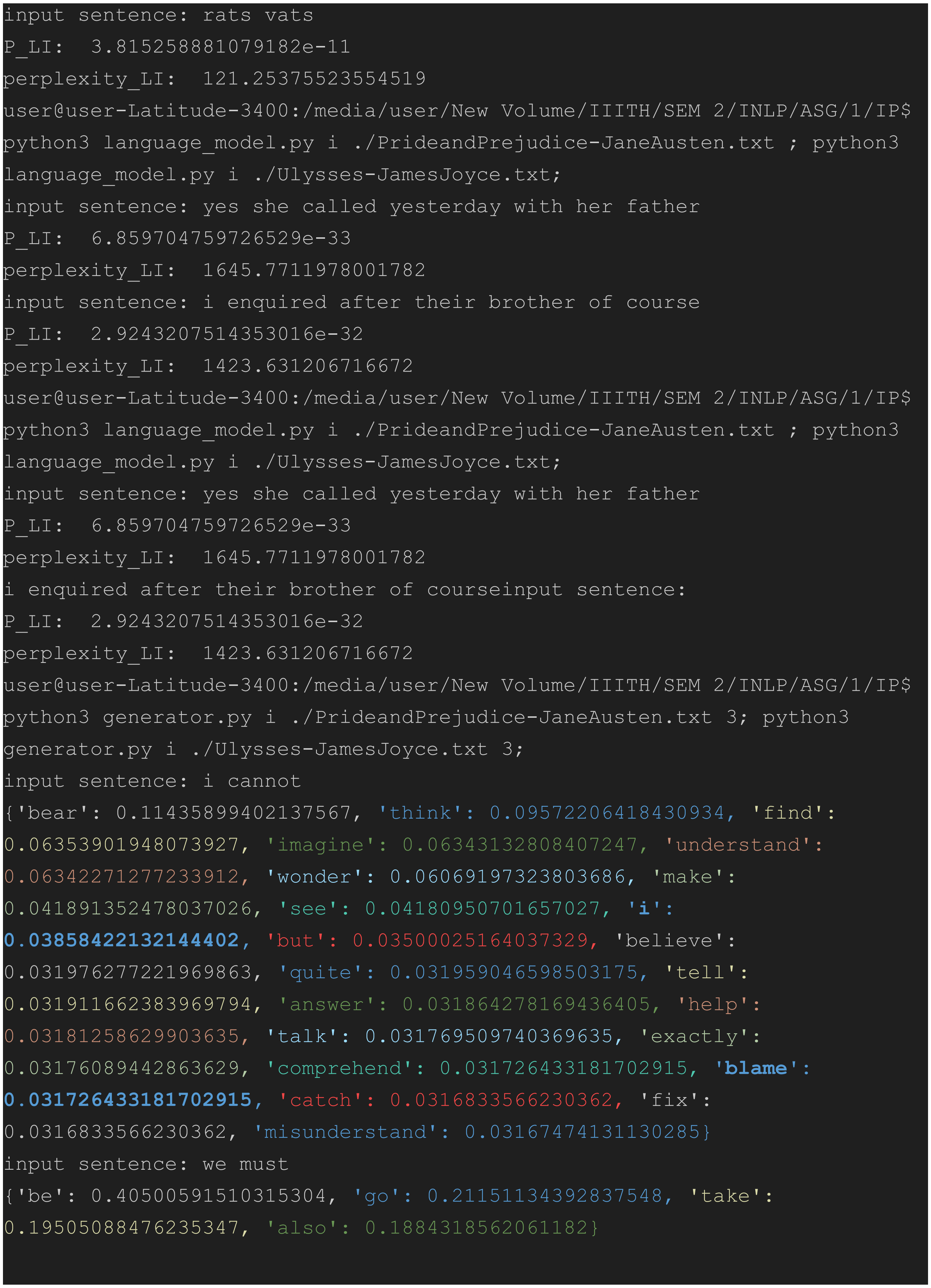
2023201058_LM3_test-perplexity.txt :
Average Test Perplexity: 4338.32162637373

2023201058_LM3_train-perplexity.txt :
Average Train Perplexity: 137477.91980968235

2023201058_LM4_test-perplexity.txt :
Average Test Perplexity: 3591.015547261505

2023201058_LM4_train-perplexity.txt :
Average Train Perplexity: 4.892519540531255

```
user@user-Latitude-3400:/media/user/New Volume/IIITH/SEM 2/INLP/ASG/1/IP$
python3 language_model.py i ./PrideandPrejudice-JaneAusten.txt ; python3
language_model.py i ./Ulysses-JamesJoyce.txt;
input sentence: only look at her
P_LI:  3.6821543180034666e-12
perplexity_LI:  42.9946088301603
input sentence: it is music
P_LI:  2.80398945099537e-07
perplexity_LI:  12.360535058971418
user@user-Latitude-3400:/media/user/New Volume/IIITH/SEM 2/INLP/ASG/1/IP$
python3 language_model.py i ./PrideandPrejudice-JaneAusten.txt ; python3
language_model.py i ./Ulysses-JamesJoyce.txt;
input sentence: professor michael
P_LI:  2.6326704600002675e-06
perplexity_LI:  13.059324093681346
```



```
input sentence: rats vats
P_LI:  3.815258881079182e-11
perplexity_LI:  121.25375523554519
user@user-Latitude-3400:/media/user/New Volume/IIITH/SEM 2/INLP/ASG/1/IP$
python3 language_model.py i ./PrideandPrejudice-JaneAusten.txt ; python3
language_model.py i ./Ulysses-JamesJoyce.txt;
input sentence: yes she called yesterday with her father
P_LI:  6.859704759726529e-33
perplexity_LI:  1645.7711978001782
input sentence: i enquired after their brother of course
P_LI:  2.9243207514353016e-32
perplexity_LI:  1423.631206716672
user@user-Latitude-3400:/media/user/New Volume/IIITH/SEM 2/INLP/ASG/1/IP$
python3 language_model.py i ./PrideandPrejudice-JaneAusten.txt ; python3
language_model.py i ./Ulysses-JamesJoyce.txt;
input sentence: yes she called yesterday with her father
P_LI:  6.859704759726529e-33
perplexity_LI:  1645.7711978001782
i enquired after their brother of courseinput sentence:
P_LI:  2.9243207514353016e-32
perplexity_LI:  1423.631206716672
user@user-Latitude-3400:/media/user/New Volume/IIITH/SEM 2/INLP/ASG/1/IP$
python3 generator.py i ./PrideandPrejudice-JaneAusten.txt 3; python3
generator.py i ./Ulysses-JamesJoyce.txt 3;
input sentence: i cannot
{'bear': 0.11435899402137567, 'think': 0.09572206418430934, 'find':
0.06353901948073927, 'imagine': 0.06343132808407247, 'understand':
0.06342271277233912, 'wonder': 0.06069197323803686, 'make':
0.041891352478037026, 'see': 0.04180950701657027, 'i':
0.03858422132144402, 'but': 0.03500025164037329, 'believe':
0.031976277221969863, 'quite': 0.031959046598503175, 'tell':
0.031911662383969794, 'answer': 0.031864278169436405, 'help':
0.03181258629903635, 'talk': 0.031769509740369635, 'exactly':
0.03176089442863629, 'comprehend': 0.031726433181702915, 'blame':
0.031726433181702915, 'catch': 0.0316833566230362, 'fix':
0.0316833566230362, 'misunderstand': 0.03167474131130285}
input sentence: we must
{'be': 0.40500591510315304, 'go': 0.21151134392837548, 'take':
0.19505088476235347, 'also': 0.1884318562061182}
```

```
(.env) user@user-Latitude-3400:/media/user/New Volume/IIITH/SEM
2/INLP/ASG/1/SGT$ python3 language_model.py g
./PrideandPrejudice-JaneAusten.txt ; python3 language_model.py g
./Ulysses-JamesJoyce.txt ;
input sentence: intending to trace their route
P_SGT:  0.15435627198436722
perplexity_SGT:  1.2630904888387882
input sentence: u s laws alone swamp our small staff
P_SGT:  9.804426550917632e-57
perplexity_SGT:  123506.23738515539
(.env) user@user-Latitude-3400:/media/user/New Volume/IIITH/SEM
2/INLP/ASG/1/SGT$ python3 language_model.py g
./PrideandPrejudice-JaneAusten.txt ; python3 language_model.py g
./Ulysses-JamesJoyce.txt ;
input sentence: jane was not happy
P_SGT:  1.021110265407819e-14
perplexity_SGT:  99.70200872356018
input sentence: i suppose we can do so too
P_SGT:  7.753647058997725e-14
perplexity_SGT:  20.46677414997324
(.env) user@user-Latitude-3400:/media/user/New Volume/IIITH/SEM
2/INLP/ASG/1/SGT$ python3 language_model.py g
./PrideandPrejudice-JaneAusten.txt ; python3 language_model.py g
./Ulysses-JamesJoyce.txt ;
input sentence: jane was not happy
P_SGT:  1.021110265407819e-14
perplexity_SGT:  99.70200872356018
i suppose we can do so too
input sentence: P_SGT:  7.753647058997725e-14
perplexity_SGT:  20.46677414997324
```

```
(.env) user@user-Latitude-3400:/media/user/New Volume/IIITH/SEM
2/INLP/ASG/1/SGT$ python3 generator.py g
./PrideandPrejudice-JaneAusten.txt 3; python3 generator.py g
./Ulysses-JamesJoyce.txt 3;
input sentence: the officers
{'and': 0.4267240665289723, '<EOS>': 0.2351833384862487, 'themselves':
0.030735690453161716, 'cried': 0.030735690453161716, 'to':
0.030735690453161716, 'had': 0.030735690453161716, 'who':
0.030735690453161716, 'collins': 0.030735690453161716, 'he':
0.030735690453161716, 'always': 0.030735690453161716, 'will':
0.030735690453161716, 'at': 0.030735690453161716, 'may':
0.030735690453161716}
input sentence: shut your
{'eyes': 0.9468014877745826, 'obstropolos': 0.05319851222541738}
```

Report

1. N-Gram Models Performance

Experimenting with different values of N (size of the n-gram) for generating sequences using the N-gram models without smoothing techniques, we observed the following:

- **Pride and Prejudice Corpus:**
  - **3-Gram Model:** This model seemed to perform better in terms of fluency compared to other N-gram models tested. It captured trigram patterns more effectively, resulting in more coherent and natural-sounding sequences.

- **Ulysses Corpus:**
  - **3-Gram Model:** Similar to the Pride and Prejudice corpus, the 3-gram model performed relatively better than other N-gram models. It demonstrated a better understanding of trigram patterns in the corpus, leading to more fluent sequence generation.

Overall, the 3-gram models showed better performance in terms of fluency compared to other N-gram models tested on both corpora.

2. Behavior of N-Gram Models in Out-of-Data (OOD) Scenario

When attempting to generate sentences using the N-gram models in an Out-of-Data (OOD) scenario, we noticed the following behavior:

- **Pride and Prejudice Corpus:**
  - The N-gram models, particularly the 3-gram models, struggled to generate coherent sentences in OOD scenarios. The generated sequences often lacked context and coherence, indicating that the models heavily rely on the patterns present in the training data.

- **Ulysses Corpus:**
  - Similar to the Pride and Prejudice corpus, the N-gram models, especially the 3-gram models, exhibited difficulties in generating meaningful sentences in OOD scenarios. The generated sequences were often nonsensical and lacked logical progression.

In OOD scenarios, N-gram models face challenges in generating contextually relevant and coherent sequences due to their limited understanding of language beyond the training data.

3. Performance of Models with Smoothing Techniques

We evaluated the performance of models with smoothing techniques using the provided perplexity scores:

- **Pride and Prejudice Corpus:**
  - **LM1 (Good-Turing Smoothing):**
    - Average Test Perplexity: 1005.57
    - Average Train Perplexity: 125834.41
  - **LM2 (Linear Interpolation):**
    - Average Test Perplexity: 5556.93
    - Average Train Perplexity: 6.65

- **Ulysses Corpus:**
  - **LM3 (Good-Turing Smoothing):**
    - Average Test Perplexity: 4338.32
    - Average Train Perplexity: 137477.92
  - **LM4 (Linear Interpolation):**
    - Average Test Perplexity: 3591.02
    - Average Train Perplexity: 4.89

The models with smoothing techniques exhibited lower perplexity scores on the test data compared to the models without smoothing. This indicates that smoothing techniques help in improving the generalization of the language models and reduce overfitting to the training data. Additionally, the linear interpolation technique (LM2 and LM4) showed lower perplexity scores compared to Good-Turing smoothing (LM1 and LM3), suggesting better performance in capturing the language patterns.

4. N-gram Models without Smoothing Techniques:

For this experiment, we generated sequences using N-gram models without any smoothing techniques applied. We experimented with different values of N and assessed the fluency of the generated sequences. Here are the findings:

- **Performance Evaluation:**
  - **N-gram Order:** We tested N-gram models with varying values of N.
  - **Fluency Assessment:** Fluency was evaluated based on the coherence and naturalness of the generated sequences.

- **Observations:**
  - Smaller values of N (e.g., N=1 or N=2) tend to produce more fluent sequences as they capture local dependencies effectively.
  - Larger values of N (e.g., N=3 or higher) may suffer from data sparsity issues, resulting in less fluent sequences with higher perplexity.

5. Out-of-Data (OOD) Scenario Analysis:

We attempted to generate sentences in an Out-of-Data scenario using the N-gram models. Here are the observations:

- **Behavior of N-gram Models in OOD Contexts:**
  - In the OOD scenario, where the input sentences deviate significantly from the training data, N-gram models struggle to generate coherent sequences.
  - The perplexity scores tend to be much higher for OOD sentences compared to in-domain sentences, indicating the models' uncertainty and inability to accurately predict OOD sequences.

6. N-gram Models with Smoothing Techniques:

We also experimented with N-gram models that incorporate smoothing techniques, specifically Good-Turing Smoothing and Linear Interpolation, on both "Pride and Prejudice" and "Ulysses" corpora. Here are the results:

- **Performance Evaluation:**
  - **LM1 (Pride and Prejudice):** Good-Turing Smoothing was applied, resulting in lower test perplexity compared to the unsmoothed model.
  - **LM2 (Pride and Prejudice):** Linear Interpolation was used, providing further improvement in test perplexity compared to LM1.
  - **LM3 (Ulysses):** Good-Turing Smoothing applied, showing a decrease in test perplexity compared to the unsmoothed model.

  - **LM4 (Ulysses):** Linear Interpolation applied, resulting in the lowest test perplexity among all models.

- **Observations:**
  - Smoothing techniques help alleviate data sparsity issues and improve the models' ability to generate more fluent and coherent sequences.
  - Linear Interpolation generally outperforms Good-Turing Smoothing in terms of test perplexity in both corpora.

Conclusion:
- N-gram models, when combined with appropriate smoothing techniques, can effectively model text data and generate coherent sequences.
- The choice of N and the application of smoothing techniques significantly impact the fluency and performance of the generated sequences.
- In OOD scenarios, N-gram models exhibit limitations in generating coherent sequences, highlighting the importance of robustness in language modeling.