Assignment #2 report

Introduction

The aim of this project is to implement a bigram n-gram language model and evaluate the perplexity of the training and test sets. This model can be used in the fields of text categorization and text checking, and also plays a role in the fields of speech input and information retrieval.

Background

N-gram Language Model: It is a statistically based language model that predicts the next word or character in a text by calculating the joint probability of n consecutive words or characters in the text.

Laplace Smoothing: also known as Add-One Smoothing, is a smoothing technique used to solve the zero probability problem in language models. In statistical language modeling, certain words or word combinations may not appear in the training data, resulting in zero probability, which affects the performance of the model. Laplace smoothing ensures that all possible events have non-zero probabilities by adding one to the count.

Approach & Challenges

Data preprocessing: read the input file, convert the text to lowercase, perform word splitting, and add special tags <s> and </s> at the beginning and end of each line, then write the processed text to the output file.

Counting words and bigrams: linearly scanning the preprocessed training set files and then counting the word frequency information and the frequency information of the diagrams.

Perform smoothing perplexity calculation: using the information obtained from the statistics in the previous step, perplexity is calculated for each sentence in the test set according to the following equation:

$$P(w_i|w_{i-1}) = \frac{count(w_{i-1}w_i)}{count(w_{i-1})}$$

$$PPL(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

In both formulas, the two count() in Eq. 1 represent the two counts obtained in the previous step, and PPL(W) represents the perplexity of the whole sentence, then in order to avoid unoccurring combinations from affecting the final results, the following formula for smoothing perplexity was used, where n is the hyperparameter and v is the total number of words in the training set:

$$P_{add-n}(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) + n}{c(w_{i-1}) + n * v}$$

Tuning parameters: the model was tried in the following intervals with different n, and finally got the best performance when n was 0.00425. The search space for n ranged from 0.000001 to 10, where tests were conducted at intervals of 1 initially. However, to achieve finer granularity and better precision, the interval was progressively refined in subsequent iterations, particularly in the range between 0.001 and 0.01. This approach allowed for a more systematic exploration of the parameter space, ensuring that the optimal value of n was identified with high accuracy. The final value of n = 0.0042 was validated through cross-validation and demonstrated consistent performance across multiple evaluation metrics, including accuracy, precision, and recall.

Results

The following result is obtained after the calculation, the lowest average perplexity is 1542.919262421487 when n is an integer, when n is 1

```
(base) D:\NLP_A2>python bigram.py -pplnb "news.test" "perplexity-n.txt" 1
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\shent\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Average Perplexity: 1542.919262421487
```

If n is not an integer, the minimum average perplexity obtained by adjusting the parameters is 413.3234044353285, when n is 0.00425

```
(base) D:\NLP_A2>python bigram.py -pplnb "news.test" "perplexity-n.txt" 0.00425
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\shent\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Average Perplexity: 413.3234044353285
```

Conclusion

In this project, we learned the significance of statistical modeling and how to implement it by completing the construction of a bigram n-gram language model。 The significance of hyperparameters for language modeling was understood in the process of tuning parameters