University of Macau

CISC3025 – Natural Language Processing

Assignment#2, 2024/2025

(Due date: **7th March**)

Problem Description

This assignment asks you to implement a bigram *n-gram language model* and evaluate the train and test set perplexity. A statistical language model computes a probability distribution over sequences of words. Given such a sequence, say of length m, it assigns a probability $P(w_1, w_2, w_3, \dots, w_m)$ to the whole sequence.

The n-gram model uses the chain rule to estimate the probability:

$$P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2|x_1), P(x_3|x_1, x_2), \dots P(x_n|x_1, \dots, x_{n-1})$$

For simplicity, we use the Markov assumption to approximate each component in the product:

$$P(w_i|w_1 w_2 \dots w_{i-1}) \approx P(w_i|w_{i-k}, \dots w_{i-1})$$

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i|w_{i-k}, \dots w_{i-1})$$

In this assignment, we use the bigram model to calculate the probability. The probability of each word is estimated based on the previous word:

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i|w_{i-1})$$

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

where $count(w_{i-1}w_i)$ corresponds to the number of times where the two words $w_{i-1}w_i$ appear jointly in the training set, $count(w_{i-1})$ corresponds to the total word frequency of w_{i-1} in the training set. To evaluate the language model, we use perplexity as the evaluation metric. For a bigram model, the perplexity is calculated as:

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

Where N is the length of the sentence, $P(w_i|w_{i-1})$ is the probability calculated from the bigram language model. To prevent the model from overfitting, several smoothing methods can be applied to the bigram language model, such as the add-1 smoothing:

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

where v is the number of words considered in your language model. More generally, the add-1 smoothing can be expanded to the add-n smoothing: (add-α smoothing in the slides)

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

Training and Test Data

The training and test data for this assignment come from the News Commentary, which is created to be used for training an English language model. The training data consists of 300 thousand lines of text. While the test set consists of around 90 thousand lines of text. The data corpora are from the official website of Shared Task: Machine Translation of News. Both the training and test data can be downloaded from UMMoodle.

Requirements

1. Write a Python program to count the words and bigrams in the corpus. The corpus is stored in a text file, each line of the file contains a sentence. The output of the program should be two files, named 'word.txt' and 'bigram.txt', where 'word.txt' contains the word count information, and the 'bigram.txt' contains the bigram count information. In the file, each line contains a word (or bigram) and its frequency, separated by a space.

```
word.txt:

apple 5
banana 10
candy 102
silly 3
the 350
.
.
.
```

```
bigram.txt:

apple tree 3
banana ball 1
i love 54
i hate 23
.
.
.
```

2. Implement a function to compute the perplexity for an input sentence using add-1 (Laplace) smoothing. The function takes a single sentence as input, and returns the computed perplexity based on the previously constructed bigram language model.

3. Extend the above function to add-n smoothing, by taking an additional parameter n in addition to the input sentence.

4. Write a function to calculate the perplexities of sentences using add-n smoothing in **batch mode** and store the results in a file.

It takes an input file ‘news.test’ and outputs a ‘perplexity-n.txt’, where ‘n’ refers to the parameter n of the add-n smoothing algorithm. The file should contain:

a) The perplexity of the whole test set, which is the averaged perplexity of all sentences.

b) The sentences and their perplexities are separated by a space.

c) **Notice** that the PPL may be INF due to the float precision problem. For these samples, output INF instead of the PPL value, and ignore them when you compute the average of the perplexities.

```
Input file:

Sentence1
Sentence2
Sentence3
…
```

```
Output file:

Test-Set-PPL
Sentence1 PPL
Sentence2 PPL
Sentence3 PPL
…
```

5. Analyze the empirical results and look for an optimal n for the test set.

6. Write a brief report to introduce your work.

The Starter Code

To make it easier for you to do this project, a starter code ‘bigram.py’ is provided. If you execute the following command under the ‘Assignment2’ directory, the program will preprocess the text file.

```
$python bigram.py -pps ‘news.train’ ‘corpus.txt’
```

Meanwhile, you can use the following command to count the word frequency:

```
$python bigram.py -cw ‘corpus.txt’ ‘word.txt’
```

The following command counts the bigrams in your provided corpus:

```
$python bigram.py -cb ‘corpus.txt’ ‘bigram.txt’
```

Moreover, you can use the following command to test your implemented add_one_perplexity() and add_n_perplexity() functions:

```
$python bigram.py -ppl1 'sentence'
```

For Requirement 4, you can run the following command to specify the name of the input and output files, as well as the add-n (i.e., $n = 1$):

```
$python bigram.py -pplnb 'news.test' 'perplexity-n.txt' 1
```

Submissions

You need to submit the following materials:

1. Runnable program and source code;
2. A brief report;
3. The output files.

Note: Please ensure that your report includes a section acknowledging the use of AI models, such as Copilot, in completing your assignment. This is required to comply with the University of Macau's policies. For more information, refer to the *Student Disciplinary Regulations of the University of Macau*.