BBM497: Introduction to Natural Language Processing Lab. **(Due: ../../..)**

Submission Assignment #1

Instructor: Necva BÖLÜCÜ *Name:* Ahmet Faruk BAYRAK, *Netid:* 21426716

1 Introduction

In this assignment, we are expected to build language models **(unigram, bigram, trigram)** and generate sentences with these models.

1.1 Dataset

As a training dataset, we used Facebook children stories dataset to build our language model. This dataset is released by Facebook that is used to train artificial intelligence software to understand children stories and predict the word that was missing from a given sentence in a story.

2 Data structures that you are used to build your language model

Our main task is building Ngram models (unigram, bigram, trigram) using the training data.

2.1 Preprocessing

But before using the training data we needed to preprocess it such as removing punctuations, removing unnecessary tokens, making all characters lower etc. And there was one specific thing: in the last line of every part there was a sentence with XXXXX and bunch of words divided by "—". I edited this sentence and replace XXXXX with the correct word. I used translate() and maketrans() function for this purpose. After editing the data I added start token "¡s¿" to the beginning of every sentence and end token "¡/s¿" to the ending of the every sentence. And finally I converted every characters to the lowercase. After that my preprocessing of the training data was done.

2.2 Building Ngrams

When preprocessing part is done building of ngrams part begins. For extracting the sentences into pairs such as binary pairs for bigram, triplet pairs for triram, I used zip() function. For every ngram there is a frequency dictionary, a model which is probability dictionary, and a smoothed model which is probability dictionary. After creating and filling the frequency dictionaries for every Ngram, creating of models part begins. In this part I used probability formulas for every Ngram and create it's model (probability dictionary).

$$p(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

Figure 1: MLE estimation

$$p(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i) + 1}{C(w_{i-1}) + V}$$

Figure 2: Add-one (Laplace) estimation

3 Error analysis of generated sentences

There is a problem I faced while I developing this program. In this assignment we create sentences by three different models: unigram, bigram and trigram. And we have three different probability and perplexity functions for each model. But at the end we call every model function for every sentences. So the problem is for example

when we call trigram probability function for the sentence which is generated from unigram model: the pairs in this sentence may not be in the trigram model. It can happen also bigram model as well. For this problem I used **smoothed values.** For smoothing I used add-one (laplace) smoothing method.

4 Calculation of perplexity

Perplexity is the inverse probability of the set, normalized by the number of words.

$$PP(W) = 2^{-\frac{1}{N}\sum_{i=1}^{N} log_2 P(w_1 w_2 \cdots w_N)}$$

After generating sentences according to three different models, program calculates the perplexity of these sentences.

```
Perplexity of unigram generated sentence:

    For unigram function: 263.4575548384702
    For bigram function: 579.6530449831633
    For trigram function: 4658.5307496917785

Perplexity of bigram generated sentence:

    For unigram function: 302.6532076696761
    For bigram function: 7.560901834419246
    For trigram function: 223.36435154804335

Perplexity of trigram generated sentence:

    For unigram function: 1509.4729955214034
    For bigram function: 38.819353868969642
    For trigram function: 3.433910417196095
```

As it shows in the output, for each model, it's function has the lowest perplexity. For unigram model generated sentences, unigram perplexity function has the lowest perplexity and trigram function has the highest perpelexity. For bigram model generated sentences, bigram perplexity has the lowest perplexity and finally for trigram model generated sentences, trigram perplexity function has the lowest perplexity.

5 Results of sentence generation

After calculating the perplexity values, the program calculates probability values of each sentences for each model functions. I set the word length to 15 and count to 2.

5.1 Unigram Model Generated Sentences

As it shown in the below, for unigram model generated sentences: unigram probability function has the highest probability and trigram probability function has the lowest probability.

```
Probability of unigram generated sentences:

Sentence: Us and be no that fairies he caps.
    For unigram function:-69.38575713429681
    For bigram function:-92.34459580388292
    For trigram function:-109.57256386643094


Sentence: When kings been are but tell aunts the the she by the for the near.
    For unigram function:-120.66385775643157
    For bigram function:-142.93591184664848
    For trigram function:-195.85301215111969
```

5.2 Bigram Model Generated Sentences

As it shown in the below, for bigram model generated sentences: bigram probability function has the highest probability.

```
Probability of bigram generated sentences:

Sentence: Do nt you have asked.
    For unigram function:-50.27498389188567
    For bigram function:-10.89939644394219
    For trigram function:-37.24173874061639

Sentence: They are very long table arrayed with kisses.
    For unigram function:-75.2058625123829
    For bigram function:-12.087787400492573
    For trigram function:-51.197938221378
```

5.3 Trigram Model Generated Sentences

As it shown in the below, for trigram model generated sentences: trigram probability function has the highest probability and the unigram probability function has the lowest probability.

```
Probability of trigram generated sentences:

Sentence: Cried the queen though she saw them.
    For unigram function:-60.20217882165798
    For bigram function:-38.53941919444228
    For trigram function:-15.409723447757015

Sentence: Some were abroad several were ill a few were in prison among the saracens others.
    For unigram function:-127.12383857356663
    For bigram function:-85.48183934972246
    For trigram function:-38.42629181433144
```

To sum up: every sentences give best result in the their generated languge model's functions. Also higher probability and lower perplexity means better results.