COSC 480C (NLP) Homework 2: Fall 2023

The due date for this homework is **Friday, October 20, 11pm ET**.

Introduction

This assignment is designed to guide you through the implementation and use of a ngram language model.

Guiding Question

- How do we assign probability to sentences?
- How can we use language models?

Learning Objectives

- Implement add-K and interpolation smoothing
- Implement information theoretic measures
- Understand how to use a language model for a task
- Understand the role of train, evaluation, and test data
- Understand how to tune hyperparameters

Rubric

1. Part 1 (165 Total Points)

Question	Points
get_ngrams	25pts
mle	25pts
addK_prob	25pts
interpolation_prob	25pts
surprisal	10pts
perplexity	10pts
entropy	10pts
byWordMetrics	10pts
generate	25pts

2. Part 2 35 Total Points

Question	Points
Tuning	15pts
Sentiment	20pts

Your assignment

Your task is to complete following steps:

1. Clone the homework repository

```
git clone https://github.com/DavisLingTeaching/HW2-ngrams.git
```

2. Install nltk version 3.8.1

```
pip install nltk==3.8.1
```

3. Download 'punkt' using python interpreter

```
>>> import nltk
>>> nltk.download('punkt')
```

4. Complete `ngram.py` described in Part 1
5. Complete `tuning.py` described in Part 2
6. Complete `classification.py` described in Part 2
7. Submit your completed programs to Moodle.

Part 1

Building a ngram model

In this portion of the assignment you will implement an ngram language model that should work with any value of n. You will add your code to `ngram.py`. You have to implement 9 functions. They are labeled with TODO in the file. A description of the function and samples of its use are given in the function docstring. **Please read the docstrings**. I have implemented some other functions that you may find useful!

Testing your ngram model

Some (non-exhaustive) tests can be run by using test.py.

For example,

```
python test.py --test ngrams
```

will test your `get_ngrams` function.

Part 2

This part should be attempted upon finishing Part 1. It is more open ended than part 1, so no supporting code is provided. Below I describe your two tasks and the data associated with them. Sample output for each is provided as a guide, but note, your results may differ slightly. You will put your code in `tuning.py` and `classification.py`, respectively.

Hyperparameter Tuning

You have implemented two types of smoothing: add-K and interpolation smoothing. These both involve hyperparameters, k and λ. Your task is to try **three different** k values and **three different lists** of λ with a trigram model. Your code should determine the best k and list of λ from those you tried. I have provided three folders of data to facilitate this, labeled `train_data`, `eval_data`, `test_data`. The file `vocab.txt` contains vocab that will serve you well in this task. **You should use the appropriate data for the task**. The final output of your `tuning.py` should be a measure of your model's performance on the relevant data.

Please include informative comments in your code and a bigger comment that describes the data you used for this task, why you used that data, and how you approached finding the best hyperparameters.

Naive Bayes Classification

In this task, you will use a unigram language model to do sentiment analysis, a common NLP task. I have provided data extracted from the Large Movie Review Dataset. The core aim of sentiment analysis is to assign a label (usually positive, negative, or neutral) to text. For this portion of the assignment you will be assigning a label of negative or positive to imdb movie reviews using Naive Bayes Classification.

Data You can find training and test data in the `sentiment_data` folder. For both train and test data, positive (`pos`) and (`neg`) reviews are provided. In `sentiment_data\train\pos`, for example, you will find a bunch of txt files which include reviews like:

```
I took part in a little mini production of this when I was a bout 8 at school
and my mum bought the video for me. I've loved it ever since!! When I was
younger, it was the songs and spectacular dance sequences that I enjoyed but
since I've watched it when I got older, I appreciate more the fantastic acting
and character portrayal. Oliver Reed and Ron Moody were brilliant. I can't
```

```
imagine anyone else playing Bill Sykes or Fagin. Shani Wallis' Nancy if the best
character for me. She put up with so much for those boys, I think she's such a
strong character and her final scene when... Well, you know... Always makes me
cry! Best musical in my opinion of all time. It's lasted all this time, it will
live on for many more years to come! 11/10!!%
```

A vocab file is provided for you, `imbd.vocab`. Use this to train your model.

Classification Naive Bayes Classification is described both in the class lecture and in Jurafasky and Martin's Speech and Language Processing 3rd Edition in Chapter 4 link. You will draw on equation (4.10):

$$c_{NB} = \text{argmax}_{c \in C} \text{log} P(c) + \sum_{i \in \text{positions}} \text{log} P(w_i|c)$$

In plain English, we can read this equation as saying, find me the class (e.g., positive, negative) which assigns the largest log probability to the document we are labeling. The probability of the document is determined by the log probability of the class (log$P(c)$) plus the total log prob of all the words in the document assigned by a model trained on examples of the relevant class.

Put another way, your tasks is to train two unigram language models, one on positive reviews the other negative reviews. In labeling test data, you gather the probability of that document with each model, weighted by the prior probability of the class. The model which assigns the highest probability (i.e. the largest log probability) is the class for that document.

Results You should report precision, recall, and f1-score for the positive and negative class test data, as well as the overall accuracy of your model. Your `classification.py` should output something like:

```
Accuracy 0.7941176470588235
Positive
    Precision 0.7830188679245284
    Recall 0.8137254901960784
    F-1 0.7980769230769231
Negative
    Precision 0.8061224489795918
    Recall 0.7745098039215687
    F-1 0.79
```

Good luck :) !