Assignment 3
Natural Language Processing
Total points: 70
Due Sunday, November 15 at 11:59 pm on Canvas

Please Work Individually

All the code has to be your own. The code is to be written individually. Violations will be reported to the University.

1. [50 points] Naive Bayes Word Sense Disambiguation

Write a Python program ***WSD.py*** that implements the Naive Bayes algorithm for word sense disambiguation, as discussed in class. Specifically, your program will have to assign a given target word with its correct sense in a number of test examples.
Please implement the Naive Bayes algorithm and cross validation yourself, do not use scikit-learn (or other machine learning library).

You will train and test your program on a dataset consisting of textual examples for the noun **"plant,"** drawn from the British National Corpus, where each example is manually annotated with its correct sense of "plant." Consider for example the following instance:

```
<instance id="plant.1000002" docsrc = "BNC/A0G">
<answer instance="plant.1000002" senseid="plant%living"/>
<context>
September 1991 1.30 You can win a great new patio Pippa Wood How to cope with a slope Bulbs
<head>plant</head> now for spring blooms
</context>
</instance>
```

The target word is identified by the SGML tag <head>, and the sense corresponding to this particular instance is that of **plant%living**.

The dataset (***plant.wsd***) is available on Canvas.

<u>Programming guidelines:</u>

Your program should perform the following steps:

- ❖ Take one argument consisting of the name of one file, which includes the annotated instances.

- Determine from the entire file the total number of instances and the possible sense labels.
- Create five folds, for a five-fold cross-validation evaluation of your Naive Bayes WSD implementation. Specifically, divide the total number of instances into five, round up to determine the number of instances in folds 1 through 4, and include the remaining instances in fold 5. E.g., if you have 122 total instances, you will have five folds with sizes 25, 25, 25, 25, and 22 respectively.
- Implement and run the Naive Bayes WSD algorithm using a five-fold cross-validation scheme. In each run, you will:

  (1) use one of the folds as your test data, and the remaining folds together as your training data (e.g., in the first run, use fold 1 as test, and folds 2 through 5 as training; etc.);

  (2) collect the counts you need from the training data, and use the Naive Bayes algorithm to predict the senseid-s for the instances in the test data;

  (3) evaluate the performance of your system by comparing the predictions made by your Naive Bayes word sense disambiguation system on the test data fold against the ground truth annotations (available as senseid-s in the test data).

Considerations for the Naive Bayes implementation:

- **All** the words found in the context of the target word will represent the features to be considered
- Address zero counts using **add-one smoothing**
- Work in **log space** to avoid underflow due to repeated multiplication of small numbers

The *WSD.py* program should be run using a command like this:
% *python WSD.py* plant.wsd

The program should produce at the standard output the accuracies of the system (as a percentage) for **each** of the five folds, as well as the average accuracy. It should also generate a file called **plant.wsd.out**, which includes for each fold the id of the words in the test file along with the senseid predicted by the system. Clearly delineate each fold with a line like this "Fold 1", "Fold 2", etc. For instance, the following are examples of lines drawn from a plant.wsd.out file

```
Fold 1
plant.1000000 plant%factory
plant.1000001 plant%factory
plant.1000002 plant%living
…
Fold 2
plant.1000041 plant%living
plant.1000042 plant%living
...
```

<u>Write-up guidelines:</u>

Create a text file called **WSD.answers**, and include the following information:

- ❖ How complete your program is. Even if your program is not complete or you are getting compilation errors, you will get partial credit proportionally. Just mention **<u>clearly and accurately</u>** how far you got.
- ❖ If your program is complete, a line consisting only of the name of the dataset: **plant.wsd**
- ❖ If your program is complete, the accuracies of your Naive Bayes system for **each** of the five folds, as well as the **average** accuracy.
- ❖ If your program is complete, identify three errors in the automatically sense tagged data, and analyse them (i.e., for each error, write **<u>one brief sentence</u>** describing the possible reason for the error and how it could be fixed)

2. [20 points] Additional Word Sense Disambiguation Experiments

Apply your Naive Bayes word sense disambiguation system on five additional datasets: ***bass.wsd, crane.wsd, motion.wsd, palm.wsd, tank.wsd***, which are also available on Canvas. As before, run your evaluation using a five-fold cross-validation. For each word (bass, crane, motion, palm, tank), your program should produce a file <word>.wsd.out, and print the accuracies for each of the folds along with the overall accuracy.

<u>Write-up guidelines:</u>

In the same text file called WSD.answers, for _each_ of the five additional datasets (*bass.wsd, crane.wsd, motion.wsd, palm.wsd, tank.wsd*) add the following information:

- ❖ How complete your program is. Even if your program is not complete or you are getting compilation errors, you will get partial credit proportionally. Just mention **<u>clearly and accurately</u>** how far you got.
- ❖ If your program is complete, a line consisting only of the name of the dataset
- ❖ If your program is complete, the accuracies of your Naive Bayes system for each of the five folds, as well as the average accuracy.

<u>There will be two links for submission on Canvas:</u>

1. Please save your full code as PDF **(plain text)** and submit it by itself.
2. Please Submit a zip file that includes all your files, ***WSD.py*** and ***WSD.answers***, but **<u>do not</u>** include the data files.
3. **Six screenshots** showing the six runs of the program (for each of the six datasets) (whether succeeded or failed), and the output or part of the output (as your screenshot allows). If a fewer number of screenshots shows the results for all datasets, it should be acceptable as well. **Please make sure the date is shown in the screenshots.**

<u>General Grading Criteria:</u>

- Submitting all required files (even with a non-compiling code): 15/70.
- A **reasonable attempt** and a reasonable code that doesn't compile: 35-40/70.
- A **<u>reasonable program/code</u>,** which runs successfully, but doesn't give the correct output: (45-50/70) (ex: few mistakes in functions or metrics, accuracy rates not rational, etc.)
- A successful program with the correct output, but not fulfilling all requirements (ex: Write-up not complete, etc.):60/70.
- A program fulfilling all the requirements: 70/70.