CSCI 544: Applied Natural Language Processing

Assignment 3: "And well you know, yet well I know"

Jonathan May (adapted from Jordan Boyd-Graber)

Out: **13. September 2017**
Due: **27. September 2017**

Introduction

The aim of this assignment is to do authorship identification on lines of poetry written by Emily Bronte and William Shakespeare. We'll be using the Naïve Bayes classifier provided by NLTK.

Unlike previous assignments, the code provided with this assignment has all of the functionality required. Your job is to make the functionality better by improving the features the code uses for text classification.

NOTE: Because the goal of this assignment is feature engineering, not classification algorithms, you *may not* change the underlying algorithm. You may wish to make modifications to the code, however, for debugging, hill climbing, resource integration, or for other reasons.

About the Data

You will be given two data files;

- `s.data` Lines of poetry by Shakespeare, marked with an 's'
- `b.data` Lines of poetry by Bronte, marked with an 'b'

The data has been downloaded from free online resources and may contain random artifacts. You are free to use that data in any way you see fit to make your features. At the very least you should use it to create a `train.tsv` file which will be used by your classifier. However, one way you might want to use the data is to divide it into train and dev subsets, in case you are worried about overfitting on the internal dev set selection.

What to Do

Apart from the data, you are provided with two files:

- `maketsv.py` : This is a convenience script for converting the provided data into a form that is suitable for the classifier.
- `classify.py` : This is an implementation of a Naive Bayes classifier that uses NLTK's Naive Bayes code. It generates features from training data, evaluates on a small percentage of the training data, then re-trains on the full training data and generates predictions for the test data. It is implemented with a baseline feature set

To begin with you should prepare the training data:

```
cat s.data b.data | python maketsv.py -o train.tsv
```

You can then run the baseline classifier:

```
python classify.py -i train.tsv
```

You should see output like:

```
Training classifier ...
Accuracy on dev: 0.777626
No test file passed; stopping.
```

Your primary job is to create additional features and raise your classifier's performance on a blind test set. If you think it is a good idea you may also augment your training data, subject to the limitations discussed below. You will be competing against your fellow students for maximum score, so apart from the normal caveats to not cheat, you may wish to avoid discussing your strategy on this homework!

External Resources

You may gather external resources to inform your features and extend your data, subject to the following caveats:

- The external resources may not consist of any written work by Emily Bronte or William Shakespeare or a derivative of said work (e.g. n-gram counts of Bronte or Shakespeare texts.) **Failure to follow this instruction will result in a submission scored equivalent to one receiving a 50% penalized score (assuming the rest of the submission is valid).**
- The external resources must be small enough to allow your code to run within the limits imposed by Vocareum. You should test this ahead of time, because a non-working submission will be scored 0.

If you wish to read in external resources, modify `classifier.py` to read these files in from your workspace in Vocareum.

Submission

You should submit:

- Your amended and suitably modified (see above) `classifier.py`.
- Your training file, `train.tsv`.
- Any external resource files that will be read by `classifier.py`.
- a file called `explanation.txt` explaining your process of creating additional features.

We will invoke your classifier as `python classifier.py -i train.tsv -t test.tsv -o answers.tsv` (we will provide `test.tsv` and collect `answers.tsv`).

The auto-grader will run your classifier on the blind test set and a score on a **portion** of this set will be calculated and returned to you. Your fellow students will also be able to see your score on this set. You may submit your homework for auto-grading up to 10 times per calendar day. Your **last** (not necessarily your best) submission will be considered your official submission.

How this Assignment is Graded (90+ points; 100 max)

You'll get 20 points for providing an adequate and clear explanation of the features you used and up to 5 additional points for innovative approaches to features and data, even if they didn't result in higher scores.

You'll get: (each item is cumulative)

- 30 points if your code produces valid output (the baseline code produces valid output so this shouldn't be a problem).
- +20 points if your code equals the baseline system performance on the **entire** blind test set (note that you will not see results on this set until after submissions have closed)
- +10 points if your code exceeds baseline system performance on the blind test set

- +10 points if your code is significantly above baseline system performance (TBD by grading staff)
- +5 points if your code is in the top 15% of class submissions
- +5 points if your code is among the very best of class submissions (TBD by grading staff)

Note this total is more than 100, however 100 is the maximum score you can receive on this assignment. If you get 90 on this assignment you should consider it a job well done.

Please remember the restrictions on data noted above and the penalties associated with violating these restrictions. Please also recall the prohibitions on plagiarizing others' code (and having your code plagiarized)