USC

CSCI 544 — Applied Natural Language Processing

Coding Exercise 2

Due: April 5, 2018, at 23:59 Pacific Time (11:59 PM)

This assignment counts for 10% of the course grade.

Assignments turned in after the deadline but before April 8 are subject to a 30% grade penalty.

Overview

In this assignment you will write a naive Bayes classifier to identify hotel reviews as either true or fake, and either positive or negative. You will be using the word tokens as features for classification. The assignment will be graded based on the performance of your classifiers, that is how well they perform on unseen test data compared to the performance of a reference classifier.

Data

A set of training and development data will be made available as a compressed ZIP archive on Blackboard. The uncompressed archive will have the following files:

- One file `train-labeled.txt` containing labeled training data with a single training instance (hotel review) per line (total 960 lines). The first 3 tokens in each line are:
  1. a unique 7-character alphanumeric identifier
  2. a label `True` or `Fake`
  3. a label `Pos` or `Neg`

  These are followed by the text of the review.
- One file `dev-text.txt` with unlabeled development data, containing just the unique identifier followed by the text of the review (total 320 lines).
- One file `dev-key.txt` with the corresponding labels for the development data, to serve as

an answer key.

- Readme and license files (which you won't need for the exercise).

Programs

You will write two programs: `nblearn.py` will learn a naive Bayes model from the training data, and `nbclassify.py` will use the model to classify new data. If using Python 3, you will name your programs `nblearn3.py` and `nbclassify3.py`. The learning program will be invoked in the following way:

```
> python nblearn.py /path/to/input
```

The argument is a single file containing the training data; the program will learn a naive Bayes model, and write the model parameters to a file called `nbmodel.txt`. The format of the model is up to you, but it should follow the following guidelines:

1. The model file should contain sufficient information for `nbclassify.py` to successfully label new data.
2. The model file should be human-readable, so that model parameters can be easily understood by visual inspection of the file.

The classification program will be invoked in the following way:

```
> python nbclassify.py /path/to/input
```

The argument is a single file containing the test data file; the program will read the parameters of a naive Bayes model from the file `nbmodel.txt`, classify each entry in the test data, and write the results to a text file called `nboutput.txt` in the same format as the answer key.

Submission

All submissions will be completed through Vocareum; please consult the instructions for how to use Vocareum.

Multiple submissions are allowed; only the final submission will be graded. Each time you submit, a submission script trains your model on the training data, runs your classifier on the development data, and reports the results. Do not include the data in your submission: the submission script reads the data from a central directory, not from your personal directory. You are encouraged to **submit early and often** in order to iron out any problems, especially issues with the format of the final output.

The performance of you classifier will be measured automatically; failure to format your output correctly may result in very low scores, which will not be changed.

For full credit, make sure to submit your assignment well before the deadline. The time of submission recorded by the system is the time used for determining late penalties. If your

submission is received late, whatever the reason (including equipment failure and network latencies or outages), it will incur a late penalty.

If you have any issues with Vocareum with regards to logging in, submission, code not executing properly, etc., please contact Siddharth.

Grading

After the due date, we will train your model on a combination of the training and development data, run your classifier on unseen test data, and compute the F1 score of your output compared to a reference annotation for each of the four classes (true, fake, positive, and negative). Your grade will be based on the performance of your classifier. We will calculate the mean of the four F1 scores and scale it to the performance of a naive Bayes classifier developed by the instructional staff (so if that classifier has F1=0.8, then a score of 0.8 will receive a full credit, and a score of 0.72 will receive 90% credit).

Notes

- **Problem formulation.** You may treat the problem as two binary classification problems (true/fake and positive/negative), or as a 4-class single classification problem. Choose whichever works better.
- **Smoothing and unknown tokens.** You should implement some method of smoothing for the training data and a way to handle unknown vocabulary in the test data, otherwise your programs won't work. The reference solution will use add-one smoothing on the training data, and will simply ignore unknown tokens in the test data. You may use more sophisticated methods which you implement yourselves.
- **Tokenization.** You'd need to develop some reasonable method of identifying tokens in the text (since these are the features for the naive Bayes classifier). Some common options are removing certain punctuation, or lowercasing all the letters. You may also find it useful to ignore certain high-frequency or low-frequency tokens. You may use any tokenization method which you implement yourselves. Experiment, and choose whichever works best.

Collaboration and external resources

- This is an individual assignment. You may not work in teams or collaborate with other students. You must be the sole author of 100% of the code you turn in.
- You may not look for solutions on the web, or use code you find online or anywhere else.
- You may not download the data from any source other than the files provided on Blackboard, and you may not attempt to locate the test data on the web or anywhere else.
- You may use packages in the Python Standard Library. You may not use any other packages.
- You may use external resources to learn basic functions of Python (such as reading and writing files, handling text strings, and basic math), but the extraction and computation of

model parameters, as well as the use of these parameters for classification, must be your own work.

- Failure to follow the above rules is considered a violation of academic integrity, and is grounds for failure of the assignment, or in serious cases failure of the course.
- We use plagiarism detection software to identify similarities between student assignments, and between student assignments and known solutions on the web. **Any attempt to fool plagiarism detection, for example the modification of code to reduce its similarity to the source, will result in an automatic failing grade for the course.**
- Please discuss any issues you have on the Piazza discussion boards. Do not ask questions about the assignment by email; if we receive questions by email where the response could be helpful for the class, we will ask you to repost the question on the discussion boards.