

## 1 Overview

In this assignment you will write a **Naive Bayes** classifier to identify hotel reviews as either truthful or deceptive, 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.

## 2 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:

- A top-level directory with two sub-directories, one for positive reviews and another for negative reviews (plus license and readme files which you won't need for the exercise)
- Each of the sub directories contains two sub-directories, one with truthful reviews and one with deceptive reviews
- Each of these sub directories contains four sub directories, called “**folds**”
- Each of the folds contains 80 text files with English text (one review per file)
- The grading script will train your model on all of the training data, and test the model on unseen data in a similar format. The directory structure and file names of the test data will be masked so that they do not reveal the labels of the individual test files

## 3 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 the directory of 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:

- The model file should contain sufficient information for nbclassify.py to successfully label new data.
- 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 the directory of the test data; the program will read the parameters of a naive Bayes model from the file **nbmodel.txt**, classify each file in the test data, and write the results to a text file called - **nboutput.txt** in the following format:

```
label_a label_b path1
label_a label_b path2
:
```

In the above format, label\_a is either “truthful” or “deceptive”, label\_b is either “positive” or “negative”, and pathn is the path of the text file being classified.

## 4 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 is invoked. The submission script uses a specific portion of the training data as development data; it trains your model on the remaining 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 should only upload your program files to Vocareum, that is `nbclassify.py` and `nblearn.py` (plus any required auxiliary files, such as code shared between the programs or a word list that you wrote yourself).

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 your 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 the TAs.

## 5 Grading

After the due date, we will train your model on the full training data (including 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 (truthful, deceptive, positive, and negative). Your grade will be based on the performance of your classifier.

## 6 Notes

- **Development data:** While developing your programs, you should reserve some of the data as development data in order to test the performance of your programs. The submission script on Vocareum will use folds 2, 3, and 4 as training data, and fold 1 as development data: that is, it will run `nblearn.py` on a directory containing only folds 2, 3, and 4, and it will run `nbclassify.py` on a directory with a modified version of fold 1, where directory and file names are masked. While developing on your own you may use different splits of the data (but to get the same results as the submission script, you'll need to use the same split). The grading script will use all 4 folds for training, and unseen data for testing
- **Problem formulation:** You may treat the problem as two binary classification problems (truthful/deceptive and positive/negative), or as a 4-class single classification problem. Choose whichever works better
- **Language Model:** You can try out various language models, like bigram or tri-gram, etc and chose the best
- **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. For example, you can use add-one smoothing on the training data, and 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 lower casing 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

## 7 Collaboration and details

- This is an individual assignment. You may not work in teams or collaborate with other students. You must be the sole author of 100%

- 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 numpy, but are **NOT** allowed to use scikit, pandas, nltk, etc
- 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

## 8 Quick Reference

- Write a script **nblearn.py** which takes argument as path to input. This should write model parameters to **nbmodel.txt** any format, should be human readable
- Write a script **nclassify.py** which takes argument as path to input of test files which have to be classified. This should write result to **nboutput.txt**. Each line in the result file should contain classification about one file in the format- **label\_a label\_b path\_to\_file**

Good Luck!!

---