CIS 521: Homework 6 [100 points]

Release Date	Tuesday, March 22, 2016
Due Date	11:59 pm on Tuesday, March 29, 2016

Instructions

In this assignment, you will answer some questions on elementary probability, then extend the spam filter implemented in the previous assignment to achieve near-perfect accuracy.

A skeleton file `homework6.py` containing empty definitions for each question has been provided. Since portions of this assignment will be graded automatically, none of the names or function signatures in this file should be modified. However, you are free to introduce additional variables or functions if needed.

You may import definitions from any standard Python library, and are encouraged to do so in case you find yourself reinventing the wheel.

It is highly recommended that you follow the Python style guidelines set forth in PEP 8, which was written in part by the creator of Python. However, your code will not be graded for style.

Once you have completed the assignment, you should submit your code and your report on Eniac using the following `turnin` command, where the flags `-c` and `-p` stand for "course" and "project", respectively.

```
turnin -c cis521 -p hw6 homework6.py homework6_report.pdf
```

You may submit as many times as you would like before the deadline, but only the last submission will be saved. To view a detailed listing of the contents of your most recent submission, you can use the following command, where the flag `-v` stands for "verbose".

```
turnin -c cis521 -p hw6 -v
```

1. Probability [20 points]

Record your answers to the following questions in the designated variables in the skeleton file.

1. **[5 points]** Determine whether the following statements about conditional probability are correct, and assign one of the Python values `True` or `False` to the indicated variables to record your answers.

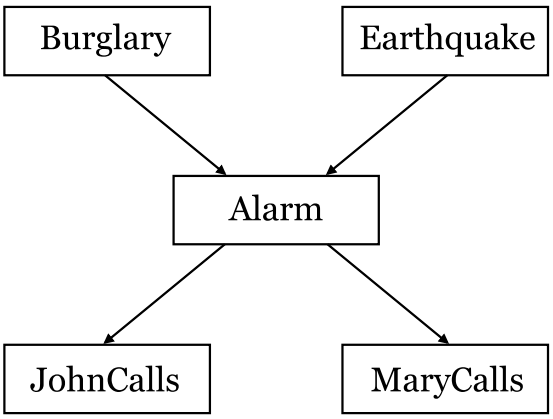
    1. If $P(a \mid b) = P(a)$, then $P(a \mid b, c) = P(a \mid c)$.

    2. If $P(a \mid b, c) = P(b \mid a, c)$, then $P(a \mid c) = P(b \mid c)$.

3. If $P(a \mid b, c) = P(a)$, then $P(b \mid c) = P(b)$.

2. **[8 points]** Consider a situation where you have an anti-burglary alarm system installed in your house. Although the alarm is set off when a burglar enters your home, it is occasionally also set off by the earthquakes that rock your neighborhood from time to time. You have asked your neighbors John and Mary to call you if they hear the alarm go off, but they might not always hear it when it sounds.

   A diagram of this situation is given below, with arrows between events corresponding to causation. Relevant probability tables are also included, with each event being abbreviated by its first letter.



$P(B)$
0.01

$P(\neg E)$
0.995

B	E	$P(A \mid B, E)$
T	T	0.97
T	F	0.95
F	T	0.23
F	F	0.001

A	$P(J \mid A)$
T	0.85
F	0.10

A	$P(M \mid A)$
T	0.95
F	0.15

   Based on this information, determine whether the following statements about dependencies between events are correct, and assign one of the Python values `True` or `False` to the indicated variables to record your solutions.

   1. JohnCalls is conditionally independent of MaryCalls, given Alarm.
   2. JohnCalls is absolutely independent of MaryCalls.
   3. MaryCalls is conditionally independent of Alarm, given JohnCalls.
   4. MaryCalls is absolutely independent of Alarm.
   5. Burglary is conditionally independent of MaryCalls, given Alarm.
   6. Burglary is absolutely independent of MaryCalls.
   7. Earthquake is conditionally independent of Burglary, given Alarm.
   8. Earthquake is absolutely independent of Burglary.

3. **[7 points]** Using the information given in the diagram above, calculate the following probabilities, and record your results in the indicated variables.

1. $P(A)$
2. $P(\neg A)$
3. $P(J)$
4. $P(E \mid A)$
5. $P(B \mid \neg A)$
6. $P(B, \neg E, \neg A, J, M)$
7. $P(\neg B, E, \neg A, \neg J, M)$

2. Spam Filter [80 points]

In the previous assignment, you implemented a baseline system for spam detection using simple single-token features in a Naive Bayes framework. Your task for this assignment is to modify and extend that system in order to improve its performance. Since the baseline classifier achieved accuracies in excess of 97% on the development set, you should aim to achieve an accuracy of 99% or higher on the development set this time around.

The only interface your code will be required to support is the following, provided in the skeleton file:

1. We should be able to initialize your spam filter using the statement `SpamFilter(spam_dir, ham_dir)`, where `spam_dir` and `ham_dir` are strings pointing to the directories containing the training spam and non-spam emails. If your classifier takes any additional parameters, such as smoothing constants, they should be hard-coded in.

2. If `spam_filter` is an instance of your `SpamFilter` class, we should be able to classify an email using the statement `spam_filter.is_spam(email_path)`, where a return value of `True` indicates that the email is spam, and a return value of `False` indicates that the email is not spam.

You are encouraged to borrow liberally from the code you wrote for the previous assignment, though some non-trivial changes will have to be made. In particular, while we will require that you maintain a Naive Bayes framework, you will want to make use of features beyond single tokens. To get you started, a few ideas are listed below, but you will also want to consider features of your own creation in order to maximize performance and robustness.

Additional features might include:

- Bigram features formed from pairs of consecutive words.
- Capitalization features, such as all-lowercase, all-uppercase, first-letter-capitalized, etc.
- Punctuation features, indicating whether a token consists of only punctuation.
- Number features, indicating whether a token is a number.
- Features related to the length of a token.

- Features derived from the email headers.

Moreover, you might also consider using tokenization methods that are more sophisticated than simply splitting on whitespace, or introducing multiple smoothing constants for different components of your system.

To help ensure that you have a comprehensive understanding of the techniques you are applying, we will require that you limit your use of external code to built-in Python modules. This excludes, for example, the NLTK platform. Before you begin writing your system extensions, think carefully about how they should be integrated into the Naive Bayes model, and what additional infrastructure they may require.

In addition to building an improved spam filter, we will also ask that you write a report documenting your findings. It should be a PDF document named `homework6_report.pdf`, and should be submitted alongside your code. Your report should at minimum include the following information:

- The general design of your system
- The features you experimented with.
- How you handled pre-processing and tokenization.
- The experiments you performed and the results you obtained, including charts or plots as needed.
- An analysis of the specific errors made by your system on the development data.

A report with the appropriate level of detail will be approximately one page in length.