CS 540-1: Introduction to Artificial Intelligence
Homework Assignment # 4

Assigned: 10/24
Due: 11/7 before class

Hand in your homework:

This homework includes a written portion and a programming portion. Please type the written portion and hand in the file in pdf format and name it as **WrittenPart.pdf**. The first page of the pdf must include a header with: **your name, Wisc username, class section, HW# , date and, if handed in late, how many days late it is**. The programming portion will be required to write in Java, and all files needed to run your program, including any support files but input/output/debug files, should be submitted. Finally, please create a folder named as <**Wisc username**>_**HW#**, put **all your codes** as well as **WrittenPart.pdf** into the folder and compress it as <**Wisc username**>_**HW#.zip**. This final zip file should then be uploaded to the appropriate place on the course Moodle website.

Late Policy:

All assignments are due at the beginning of class on the due date. One (1) day late, defined as a 24-hour period from the deadline (weekday or weekend), will result in 10% of the total points for the assignment deducted. So, for example, if a 100-point assignment is due on a Wednesday 9:55 a.m., and it is handed in between Wednesday 9:55 a.m. and Thursday 9:55 a.m., 10 points will be deducted. Two (2) days late, 25% off; three (3) days late, 50% off. No homework can be turned in more than three (3) days late. Written questions and program submission have the same deadline. A total of two (2) free late days may be used throughout the semester without penalty.

Assignment grading questions must be raised with the instructor within one week after the assignment is returned.

Collaboration Policy:

You are to complete this assignment individually. However, you are encouraged to You are to complete this assignment individually. However, you are encouraged to discuss the general algorithms and ideas with classmates, TAs, and instructor in order to help you answer the questions. You are also welcome to give each other examples that are not on the assignment in order to demonstrate how to solve problems. But we require you to:

- not explicitly tell each other the answers
- not to copy answers or code fragments from anyone or anywhere
- not to allow your answers to be copied
- not to get any code on the Web

In those cases where you work with one or more other people on the general discussion of the assignment and surrounding topics, we suggest that you specifically record on the assignment the names of the people you were in discussion with.

Question 1: Probability [20]

The following two tables provide the full joint distribution for three boolean variables X, Y and Z. Here, X indicates X =T and $\neg X$ indicates X =F. The same is also true for the other variables.

Z

	Y	$\neg Y$
X	0.08	0.10
$\neg X$	0.09	0.25

$\neg Z$

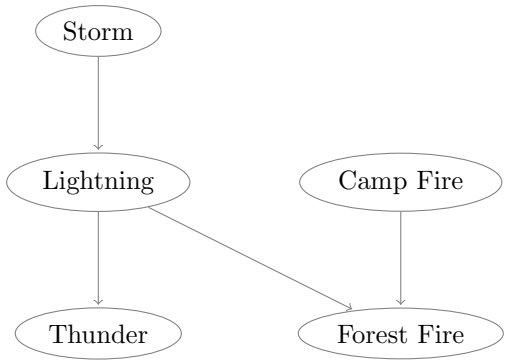
	Y	$\neg Y$
X	0.04	0.18
$\neg X$	0.09	0.17

For all of these following questions, to receive full credit, please show your work.

a) [4] What is the value of $P(X)$?
b) [4] What is the value of $P(\neg X|Y)$?
c) [4] What is the value of $P(\neg Y|X, \neg Z)$?
d) [4] Verify whether X and Z are conditionally independent given Y.
e) [4] Verify whether X and Y are conditionally independent given Z.

Question 2: Directed Graphical Model [30]

Consider the following Bayes Net over five boolean-valued random variables.



P(Storm=T)	P(Storm=F)
0.2	0.8

P(Camp Fire=T)	P(Camp Fire=F)
0.6	0.4

Storm	P(Lightning=T | Storm)	P(Lightning=F | Storm)
T	0.7	0.3
F	0.05	0.95

Lightning	P(Thunder=T | Lightning)	P(Thunder=F | Lightning)
T	0.94	0.06
F	0.25	0.75

Lightning	Camp Fire	P(Forest Fire=T | Lightning, Camp)	P(Forest Fire=F | Lightning, Camp)
T	T	0.6	0.4
T	F	0.5	0.5
F	T	0.15	0.85
F	F	0.01	0.99

For all of these following questions, to receive full credit, please show your work.

a) [6] What is the probability of thunder?
b) [6] What is the probability of thunder given a forest fire?
c) [6] What is the probability that there is a storm given that there is thunder?
d) [6] What is the probability of camp fire given that there is no thunder?
e) [6] What is the probability of a lightning and a forest fire?

Question 3: Programming Naive Bayes Classifiers [50]

One of the famous applications of Naive Bayes classifiers is in spam filtering for e-mail. Can this machine learning method be effective in classifying shorter documents, namely SMS messages (text messages) as well? You are about to find out.

In this question, your task is to implement a Bayes classifier to identify SPAM from HAM. The data set we provide you consists of 5,580 text messages that have been labelled as either SPAM or HAM and we have split it into training set and testing set . If you open up the files, you will see that the first word per line is the label while the remainder of the line is the text message.

Methods to implement

We have provided code for you that will open this file, parse it, pass it to your classifier, and output the results. What you will have to focus on is the implementation of the file `NaiveBayesClassifierImpl.java`. If you open that file, you will see the following four methods which you must implement:

- `void train(Instance[] trainingData, int v)`
  This method should train your classifier with the training data provided. The integer argument v is the size of the total vocabulary in your model. Please take this argument and store it as a field, as you will need it in computing the smoothed class-conditional probabilities. [See the section on **Smoothing** below]
- `double p_l(Label label)`
  This method should return the prior probability of the label in the training set. In other words, return $P(Spam)$ if `label == Label.SPAM` or $P(Ham)$ if `label == Label.HAM`.
- `double p_w_given_l(String word, Label label)`
  This method should return the class-conditional probability of label. In other words, return $P(word|label)$. To compute this probability, you will use smoothing. Please read the note on how to implement this below.
- `ClassifyResult classify(String[] words)`
  This method return the classification result for a single message.

We have also defined three class types in assist of your implementation. The `Instance` class is a data structure holding the label and the SMS message as an array of words:

```
public class Instance {
    public Label label;
    public String[] words;
}
```

The `Label` class is an enumeration of our class labels:

```
public enum Label { SPAM, HAM }
```

The `ClassifyResult` class is a data structure holding a label and two log probabilities (whose values are described in the log probabilities section):

```
public class ClassifyResult {
    public Label label;
    public double log_prob_spam;
    public double log_prob_ham;
}
```

The only provided file you are allowed to edit is `NaiveBayesClassifierImpl.java`. But you are allowed to add extra class files if you would like.

Smoothing

There are two concepts we use here:

- Word token: occurrences of a word.
- Word type: unique word as a dictionary entry.

For example, "the dog chases the cat" has 5 word tokens but 4 word types; there are two tokens of the word type "the". Thus, when we say a word "token" in the discussion below, we mean the number of words that occur and **NOT** the number of unique words. As another example, if an SMS message is 15 words long, we would say that there are 15 word tokens. And if the "lol" appeared 5 times, we would say there were 5 tokens of "lol".

The class-conditional probability $P(w|l)$ where w represents some word token and l a label is a multinomial random variable. If there are $|V|$ possible word types that might appear, imagine a die with $|V|$-sided die. $P(w|l)$ is just the likelihood that this die lands with the w-side up. You will need to estimate two such distributions: $P(w|Spam)$ and $P(w|Ham)$.

One might think that we would estimate the value of $P(w|Spam)$ by simply counting the number of w tokens and dividing by the total number of word tokens in all messages in our training data labelled as Spam, but this does not prove to be desirable in all cases, for the reason given below.

One complication for a Naive Bayesian classifier is the presence of unobserved events in test data. Lets be concrete about this in the case of our classification task. Consider the word pale. Lets say pale does not appear in our training data but occurs in our test data. What probability would our classifier assign to $P(pale|Spam)$ and $P(pale|Ham)$? The probability would be zero, and because we are taking the sum of the log of the the conditional probabilities for each word and $\log 0$ is undefined, the expression whose maximum we are computing would be undefined. This would be a rather unhelpful classifier.

What we do to get around this is that we pretend we actually did see some (possibly fractional many) tokens for the word type pale. This goes by the name of Laplace smoothing or add-δ smoothing, where δ is a parameter. We write:

$$P(w|l) = \frac{C_l(w) + \delta}{|V|\delta + \sum_{v \in V} C_l(v)}$$

with $l \in \{Spam, Ham\}$, $C_l(w)$ representing the number of times the token w appears in messages labeled l in the training data. As above, $|V|$ represents the size of the total vocabulary we assume we will encounter. Thus it forms a superset of the words used in the training and test sets. The value $|V|$ will be passed to the train method of your classifier as the argument int v. For this assignment, please use the value $\delta = 0.00001$. With a little reflection, you will see that if we estimate our distributions in this way, we will have $\sum_{w \in V} P(w|l) = 1$.

Please use this equations above for $P(w|l)$ to calculate the class-conditional probabilities in your implementation.

Log probabilities

The second gotcha that any implementation of a Naive Bayes classifier must contend with is underflow. Underflow can occur when we take the product of a number of small floating-point values. Fortunately, there is a workaround.

Recall that a Naive Bayes classifier computes:

$$f(w) = \arg\max_l \left[P(l) \prod_{i=1}^{k} P(w_i|l) \right]$$

where $l \in \{Spam, Ham\}$ and w_i is the i^{th} word token of your SMS message, numbered 1 to k. Because maximizing a formula is equivalent to maximizing the log value of that formula, $f(w)$ computes the same class as:

$$g(w) = \arg\max_l \left[\log P(l) + \sum_{i=1}^{k} \log P(w_i|l) \right]$$

What this means for you is that in your implementation of classify, compute the $g(w)$ formulation of the function above rather than the $f(w)$ formulation. This will result in quicker code and avoid errors generated by multiplying small numbers. Note however, your methods `p_l` and `p_w_given_l` should return the true probabilities themselves and **NOT** the log of the probabilities.

This is what you return in the `ClassifyResult` class: `log_prob_spam` is the value $\log P(l) + \sum_{i=1}^{k} \log P(w_i|l)$ with $l = Spam$ and `log_prob_ham` is the value with $l = Ham$. The label returned in this class corresponds to the output of $g(w)$.

How to test

We will test your program on multiple training/tuning/testing cases, and the format of testing commands will be like this:

```
java HW4 trainingFilename testFilename
```

which trains the classifier from the data in `trainingFilename` and tests it against `testFilename`, outputting the classifiers prediction for the label and the true label for each instance in the testing data, in the same order as it occurs in `testFilename`.

In order to facilitate your debugging, we are providing you with sample input files and the corresponding output files. They are `train0.txt` and `test0.txt`, as seen in the zip file. So here is an example command:

```
java HW4 train0.txt test0.txt
```

As part of our testing process, we will unzip the file you return to us, remove any class files, call `javac HW4.java` to compile your code, and call the main method of HW4 with parameters of our choosing. Make sure that your code runs on one of the computers in the department because we will conduct our test on such computers.

Deliverables

1. Hand in (see the cover page for details) your modified version of the code skeleton we provide you with your implementation of the Naive Bayesian classifier. Also include any additional java class files needed to run the program.
2. Optionally, in the written part of your homework, add any comments about the program that you would like us to know.