CSC403: Final Project

Write a Bayesian classifier.

Your program should consist of one file `Classifier.java` deifining one public class `Classifier` with at least three methods:

A method `learn(filename)` that reads an arbitrary piece of text from the file `filename`, for example "the cat is on the roof and the cat is black", and builds graph where every vertex contains a unique word (*the, cat, is, on, roof, and, black*) connected directed links storing a counter of how many times that word follows the other. For example: (*cat* follows *the* in 2 times, *roof* follows *the* in 1 time, *is* follows *cat* in 2 times, *on* follows *is* 1 time and *black* follows *is* also 1 time, etc). A word is a vertex and a counter is a link. A lack of link connecting two vertices indicates that those words do not apper one after the other.

A method `mumble(n)` that starts from a random vertex in the graph and builds a random path (of length n) starting from that vertex in such a way that from each visited vertex, it moves on to a next vertex with a probability proportional to the counter connecting the links. For example if you start at *the* and *cat* follows *the* 2 times while *roof* follows *the* once, then you move from *the* to *cat* with a 66% probability and from *the* to *root* with a 33% probability. The method `mumble(n)` should return a string containing the words representing the vertices in the path. Possible output include:

- "the cat is on the roof"
- "the cat is on the cat"
- "the roof and the cat'

but should not include:

- "the roof is the cat"

A method `main` that takes a filename as input, `learn`s form that file, and `mumble`s some random text of the same length.

- The code should be indented.

- The code should be documented (for each functions there should be an explanation of the purpose of the function, its input, and its output with an example).
- Include a short paper (4-6 pages) explaining the running time of the `learn` algorithm and the `mumble` algorithm, and examples.
- You should try to make your code as fast as possible (for example you should avoid searching the graph for words but use a binary search tree or a hash table to locate words in the graph).

You will receive up to 10points for working code. Up to 10points for documentation. Up to 10 points for your discussion of the algorithms running time.