PS6

MARKOV MODEL OF NATURAL LANGUAGE

In this assignment, we analyze an input text for transitions between *k-grams* (a fixed number of characters) and the following letter.

Then we produce a probabilistic model of the text: given a particular *k-gram* series of letters, what letters follow at what probabilities?

Then we use the model to generate nonsense text that's surprisingly reasonable.

See the full assignment at the Princeton site:
http://www.cs.princeton.edu/courses/archive/spr15/cos126/assignments/markov.html

```
MarkovModel(string text, int k) // create a Markov model of order k
from given text
```



```
                                // Assume that text has length at
least k.
int order()                     // order k of Markov model
int freq(string kgram)          // number of occurrences of kgram in
text
                                // (throw an exception if kgram is not
of length k)
int freq(string kgram, char c) // number of times that character c
follows kgram
                               // if order=0, return num of times char
c appears
                                // (throw an exception if kgram is not
of length k)
char randk(string kgram)        // random character following given
kgram
                               // (Throw an exception if kgram is not of
length k.
                               //  Throw an exception if no such kgram.)
string gen(string kgram, int T)  // generate a string of length T
characters
                              // by simulating a trajectory through the
corresponding
                              // Markov chain.  The first k characters
of the newly
                              // generated string should be the argument
kgram.
                              // Throw an exception if kgram is not of
length k.
```

DETAILS

- This is the same as the Princeton spec except that the class method `rand` has been renamed `randk`.
- Here is a header file to get you started: MarkovModel.hpp.
- Make sure you throw `std::runtime_error`'s in the methods per the spec. Review how to do this in PS5a.
- Test your implementation against the following test file: mmtest.cpp.
- Consider the behavior of a 0-order model. This model will generate new characters with a distribution proportional to the ratio they appear in the input text. This model is context-free; it does not use an input kgram (representing the current state of the model) when generating a new character. As such:
  - The `freq(string kgram)` method takes an empty string as its input kgram (length of kgram must equal the order of the model).
  - This method call should produce as a result the length of the original input text (given in the constructor).
  - The `freq(string kgram, char c)` also will take a null string as input. It should produce as output the number of times the character `c` appears in the original input text.
- Per discussion in class, consider using a C++ map (http://www.cplusplus.com/reference/map/map/) store frequency counts of each kgram and ``k+1''-gram as it's encountered when you traverse the string in the constructor.
- Make sure to wrap around the end of the string during traversal, as described in

these sample instructions for how to do the overload. You should print out all of the *k*-gram and *k+1*-gram frequencies, the order of the model, and its alp . Here are sample instructions for how to build an iterator over the map of *k*-gram and *k+1*-gram keys.

- Create a Makefile
- Create a ps6-readme.txt.

SUBMIT

Submit the following:

- your code files `MarkovModel.cpp` and `MarkovModel.hpp`
- a source file `TextGenerator.cpp` for the Text generation client per the Princeton spec
- your Makefile which must build an executable named `TextGenerator`
- your `ps6-readme.txt` file

Submit using the submit utility as follows:

submit schakrab ps6 ps6

GRADING RUBRIC

MM implementation: 4

(full & correct implementation=4 pts; nearly complete=3pts; part way=2 pts; started=1 pt)

Readme: 2

(discussion is expected -- at least mentioning something per section.)

Total: 12