Intermediate Report:

Vivian Tan (vyt73)

The overall project is based on applications of language models, more specifically text generation based on n-grams. Random text generation can be useful for designers to use in different settings such as web applications, site templates, and topography demos.

I will be using the n-gram language model to complete this project, starting with bigrams and then moving to trigrams if time permits. I will also use Maximum Likelihood Estimation (MLE) to calculate the probabilities used to select subsequent words. To store the texts, I will be using lists of strings/tokens for each work. Eventually, I hope to combine different works from the same author into a single list.

I'll use texts from Project Gutenberg as my corpus, more specifically, from these Authors: (may remove/add authors based on progress)

- Jane Austen
- G.K. Chesterton
- William Shakespeare
- Arthur Conan Doyle
- Fyodor Dostoyevsky

Below I will discuss my initial results.

Initial results:

This project is based on applications of language models, more specifically text generation based on n-grams. The end goal is to generate a random text based on two different author's writing styles, but for this initial report we will:

- explore different methods to generate a random text from a single work
- explore different techniques to get works from URLs

Generating text:

First, let's generate a random text from a single work:

Using bigrams and MLE by hand: (based on hw code)

This first technique is from one of the homework assignments. It generates text by counting the bigrams for a text and calculating the probabilities for each then using these to generate the next word. It could easily be modified to account for trigrams. I tested out the text generation with a text from the Gutenberg library, "Emma". The results show that we can generate a fairly coherent paragraph based on a single text using this technique. Let's try another technique:

Generating text using nltk:

Based on: http://www.katrinerk.com/courses/python-worksheets/language-models-in-python (http://www.katrinerk.com/courses/python-worksheets/language-models-in-python)

This technique requires less lines of code since it just uses some built-in nltk functions. However, this method is limited to only bigrams.

This technique can also generate a paragraph based on a single text, but it doesn't quite seem as cohesive as the previous generated paragraph. It's sufficient for our purposes though. I think I may use the 2nd technique if I want to test text generation quickly and use the 1st technique with trigrams for the final result.

Now that we know that we can generate text from a single work, the next step would be to generate a single text from multiple works. To do that, we need to know how to pull multiple works from online sources. Next we will explore different techniquess to get texts online.

Getting multiple works online:

Through the nltk corpus:

The nltk gutenberg corpus comes with a few works, but it doesn't have all the authors I want or enough works for each author (probably want at least 5 for each).

However, if we decide to scale our project down to these works, we could save each of the texts and then concatenate them later. Let's see if we can get books directly from the Gutenberg website:

Through URLS:

To get text through URLs, I used the urllib library to retrieve data from urls. I was able to get a text file using the url successfully using this library.

I could also successfully generate a random text from a file pulled from a url, which will be useful in future work.

Conclusion

In this intermediate report we were able to:

- Generate random text from a single file using 2 techniques
- Retrieve works from the nltk Gutenberg corpus
- Retrieve a text file from a URL and generate a text from it

The next steps will be:

- Retrieve multiple texts from URLs
- Combine multiple books for each author
- Generate a text based on two texts/authors