Petty Authors

Authors Stoker and Gilman want to sit with the classically cool Lady Shelly. To decide who she'll allow to sit with her, these authors most famous works, Dracula, The Yellow Wallpaper and Frankenstein respectively, were compared using an ngram model.

To achieve this, the books were downloaded from project gutenburg and cleaned of licenses and added info not written by the authors. Because one of the aims was to construct plausible sentences they might have spoken, the books could not merely be word tokenized. This would not include any start or stop tags to tell the model about the structure of a sentence. Therefore the books were first sentence tokenized using nltk. The last pre-processing step was to lowercase the strings and word tokenize the sentences.

One measure of similarity between texts is the number of common ngrams shared. Initially bigrams were targeted. A pandas 2d dataframe with the lexicon as indices was used to count the number of occurrences for each bigram. Unfortunately my laptop was unable to run the calculation in a reasonable period of time (< 10 minutes) even for a 4th, smaller work. To get around this problem a simple list of tuples of neighboring words was created. Then the list was filtered by a minimum_frequency in the text (generally 15-20 based on the text's size). This produced so many matches that a move up to trigrams was needed.

The process of generating a list of frequently found trigrams was essentially the same, however more meaningful frequencies were found. The list of frequent trigrams for each book was then converted to a set and the intersection was taken with that of another book to find the overlapping frequently found trigrams.

The results were as follows:

Total Frequent Trigrams :

F:235

D:1059

Y:31

Trigrams in common between:

Frankenstein and Dracula: 127 or 9.8145 %

Frankenstein and The Yellow Wallpaper: 5 or 1.8797 %

Dracula and The Yellow Wallpaper: 17 or 1.5596 %

This showed that Shelly would have a clear preference for Stoker.

The odd one out, Gilmore did not have very much in common with either, but would narrowly choose Shelly over Stoker.

To produce a possible conversation between Shelly and Stoker, a different architecture was needed. Upon further investigation, the

```
df.loc('row', 'col')
```

function was the bottleneck when using a dataframe to produce a probability table and a much more optimized, though lesser known alternative is:

```
df.at()
```

Swapping this access method out meant that a count table was possible for my machine to run, though very slow. Once counts were made, smoothing was done by adding 0.001 to all values. This is essentially Laplace Smoothing, but skews the resulting probabilities far less than adding 1.0 to all values. Empirically the probability of “a” given start_token changed from 13.6% to 13.1%, a relatively small change.

With smoothing done, the probability for all combinations was calculated by summing each row and dividing the entries by the result. Then a function to choose a word given the probability table and a previous word was implemented by

1.) Taking the row of probabilities indexed by the previous word
2.) Choosing a random value between 0 and 1
3.) Walking through the row and subtracting each value from the random value until it was below 0
4.) Index the list of words to find the selected next word

This process was repeated until an end_token was produced to generate the conversation.