DS 2501: Intermediate Programming with Data / Lab Practicum
Prof. Rachlin, Northeastern University

Hambot

In natural language processing, n-grams refer to all of the n-length contiguous words occurring over the text. For example, if our input is: "*She sells sea shells by the sea shore*" the 2-grams would be (ignoring case): she sells, sells sea, sea shells, shells by, … etc. and our 3-grams would be: she sells sea, sells sea shells, sea shells by, shells by the, … etc.

We're going to try to use 2-grams and 3-grams to create a lyrics-generating bot for the musical *Hamilton*.

Here's the general approach:

1. Read the file containing Hamilton lyrics. Remove punctuation, headers that indicate the speaker such as **[HAMILTON]** or **[HAMILTON & LAURENS]** and convert all words to lowercase.
2. Catalog all the words that begin a sentence.
3. Catalog all the words that terminate a sentence
4. Catalog all the 2-grams as a dictionary where the key is the first word, and the value is a *list* of all the possible follow-on words. For example, in the tongue twister provided above we might have:

   start_words: [she]
   stop_words: [shore]
   2-grams: {she: [sells], sells: [sea], sea:[shells, shore], … }.

5. Now generate random Hamilton lyrics by doing the following:
   a. Pick a random start word
   b. For each current word, pick a random next word from your 2-gram dictionary by looking up the possible follow-on words and choosing one at random.
   c. If the chosen word is a stop word, your sentence is done, otherwise go to step b.

6. Does this produce anything reasonable or complete gibberish? I suspect maybe the latter. Submit your code and your best lyrics.

7. **To earn a 5 on the lab**: Modify your code to work with 3-grams. Now the start and stop words become start / stop word pairs, and the 3 grams map pairs of words (tuples) to lists of possible 3rd words. This should produce sentences that are more coherent. Does it work?