Machine Learning & Data Mining | Caltech CS/CNS/EE 155
Miniproject 2 | Released February 17th, 2017

SONNET SPOOFER

by Shari Kuroyama and Carly Robison

1 Tokenizing

Methods Used

What methods did you use and try to tokenize the sonnets?

- First we tried the naïve tokenization of using each word as a token. With enough states and iterations, this actually learned fairly well.
- We trained on individual lines from the poems. When we needed to generate the lines backwards from the rhymes at the end, we trained on individual lines read in backwards.
- Words were extracted by splitting on the spaces between them, so hyphenated words were included and every different spelling/conjugation of a word was counted separately.
- We kept apostrophes in words, but we removed periods, commas, colons, and parentheses.

Modifications

Did you have to make changes to the way you tokenized after running the algorithm and seeing the results? **Short answer: No.**

- We thought about splitting each word into syllables in order to keep to the 10-syllable format of a sonnet. However, we got around that by counting the number of syllables generated per line, and generating new words if the last word was too long.
- We also thought about trying to train using bigrams of words as tokens; but again, the naïve implementation worked fairly well.
- The naïve implementation was okay at getting meter, so we didn't see the need to supervise training to improve the meter.

2 Algorithm

Packages Used

We used the HW5 solutions for unsupervised HMM training. We used `NLTK`'s dictionary of words to get syllable counts for most words and the `pronouncing` package, which is based on `NLTK`'s `CMUdict`, to find rhymes. For words that were not in `NLTK`'s `CMUdict`, we used the count_syllables function from a package called Poetry-Tools[1].

Parameters Used

- **Hidden States.** We experimented with different numbers of hidden states. For testing our algorithms we started with 5 hidden states, which produced poems with no sense at all. We found that the poems generated with 10 or 20 states were about the same level of coherence.
- **Number of Poems.** With fewer poems, we got more content coherence; but with more training data the poems we generated had more grammatical sense.
- **Number of Iterations.** We found it hard to tell how many iterations to use; we used anywhere from 15 to 100 for our final poems, and this choice was almost exclusively based on runtime.

[1]https://github.com/hyperreality/Poetry-Tools/blob/master/poetrytools/countsyl.py

SONNET SPOOFER
by Shari Kuroyama and Carly Robison

3 Poetry Generation

Generation Process

We iterated through multiple ideas to get a good sonnet. We first took 14 emissions of 10 lines each. We added rhyming by pre-generating pairs of rhyming words that both occurred in Shakespeare's sonnets, and adding them to the end of the lines in the `ABABCDCDEFEFGG` rhyming scheme. We eventually succeeded in "seeding" the HMM emission process with a hidden state chosen according to the emission probabilities for a given word. Our final poetry generation process trained on backwards lines and generated lines backwards from the rhyme at the end. We reversed the line and combined them to make a 14-line sonnet.

How to get it to look like a sonnet?

We used `pronouncing`'s functionality to ensure the correct rhyme scheme. We ensured 10 syllables by predicting sequences of new words when we ran out of space, and doing so until we could finish the line. We generated 14 lines.

Do they make sense?

We did not train or enforce meter, yet many of the sonnets have some kind of iambic meter, but not always carried through all of the lines. It is hard to tell from the most popular words for each state, but some states must have contained words which started with stressed syllables and others would not have, and alternating between states would give the correct iambic stress pattern. The rhyme and syllable counts are correct, because our generation process enforces their correctness. Most of the sonnets make sense if read dramatically, and because of the vocabulary they still sound like Shakespeare.

0

Seasons him make but heart my such being,
Rosy gardens all that although is morn,
Serving against self beauty i seeing,
Their name very so for and proof thy mourn,
Of by walls till that touches to the fixed,
In lawful who perceived cures whom and page,
All to all weed absent better betwixt,
Thy please should mayst hold outbraves to age,
And you which my us truly in art you,
Shall whose shapes but his and with my straying,
Which so dearest temptation i woe through,
What to that it cross not in be saying,
Black enough make muse thee like of thee halt,
Of not eyes flies false the viewest did fault.

Figure 1: Submitted poem, generated with 40 states. Since Shakespeare's sonnets are numbered 1 to 154, and because computer scientists zero-index, we numbered our sonnet 0.

4 Visualization and Interpretation

Hidden State Meaning

```
##########################################################################################
            Most likely words for each state. 10 states 20 iters
##########################################################################################
state 0    words: ['for', 'and', 'with', 'that', 'when', 'to', 'which', 'by', 'where',
'how', 'then', 'let', 'o', 'within', 'so', 'all', 'why', 'like', 'in', 'as']
state 1    words: ['of', 'love', 'self', 'in', 'heart', 'eye', 'eyes', 'have', 'to',
'from', 'own', 'on', 'am', 'beauty', 'not', 'no', 'world', 'sweet', 'will', 'doth']
state 2    words: ['the', 'thy', 'my', 'with', 'to', 'it', 'or', 'they', 'mine', 'his',
'i', 'and', 'that', 'thou', 'your', 'a', 'have', 'some', 'dost', 'when']
state 3    words: ['to', 'of', 'a', 'thou', 'me', 'in', 'that', 'than', 'what', 'her',
'all', 'such', 'will', 'thy', 'as', 'can', 'but', 'this', 'doth', 'thee']
state 4    words: ['in', 'all', 'not', 'to', 'that', 'this', 'on', 'i', 'so', 'from', 'he'
'are', 'what', 'like', 'as', 'is', 'art', 'which', 'me', 'for']
state 5    words: ['and', 'but', 'so', 'as', 'that', 'nor', 'then', 'by', 'thou', 'which',
'or', 'how', 'when', 'thus', 'for', 'if', 'o', 'seem', 'not', 'yet']
state 6    words: ['is', 'be', 'of', 'that', 'thee', 'more', 'do', 'love', 'now', 'and',
'praise', 'hath', 'their', 'have', 'are', 'still', 'doth', 'being', 'with', 'live']
state 7    words: ['i', 'not', 'thee', 'me', 'that', 'time', 'o', 'she', 'be', 'you',
'art', 'him', 'it', 'truth', 'self', 'love', 'is', 'he', 'day', "time's"]
state 8    words: ['thou', 'and', 'if', 'yet', 'but', 'to', 'for', 'who', 'when', 'that',
'those', 'no', 'so', 'as', 'with', 'which', 'can', 'do', 'therefore', 'by']
state 9    words: ['my', 'the', 'i', 'thy', 'to', 'a', 'your', 'his', 'thee', 'you', 'this
'it', 'mine', 'thine', 'should', 'me', 'of', 'their', 'sweet', 'one']
```

Thanks to the team Hidden Statespeare, we found this tool for visualizing our transition matrices. Visualized Transition Matrix, 10 states, 20 iters

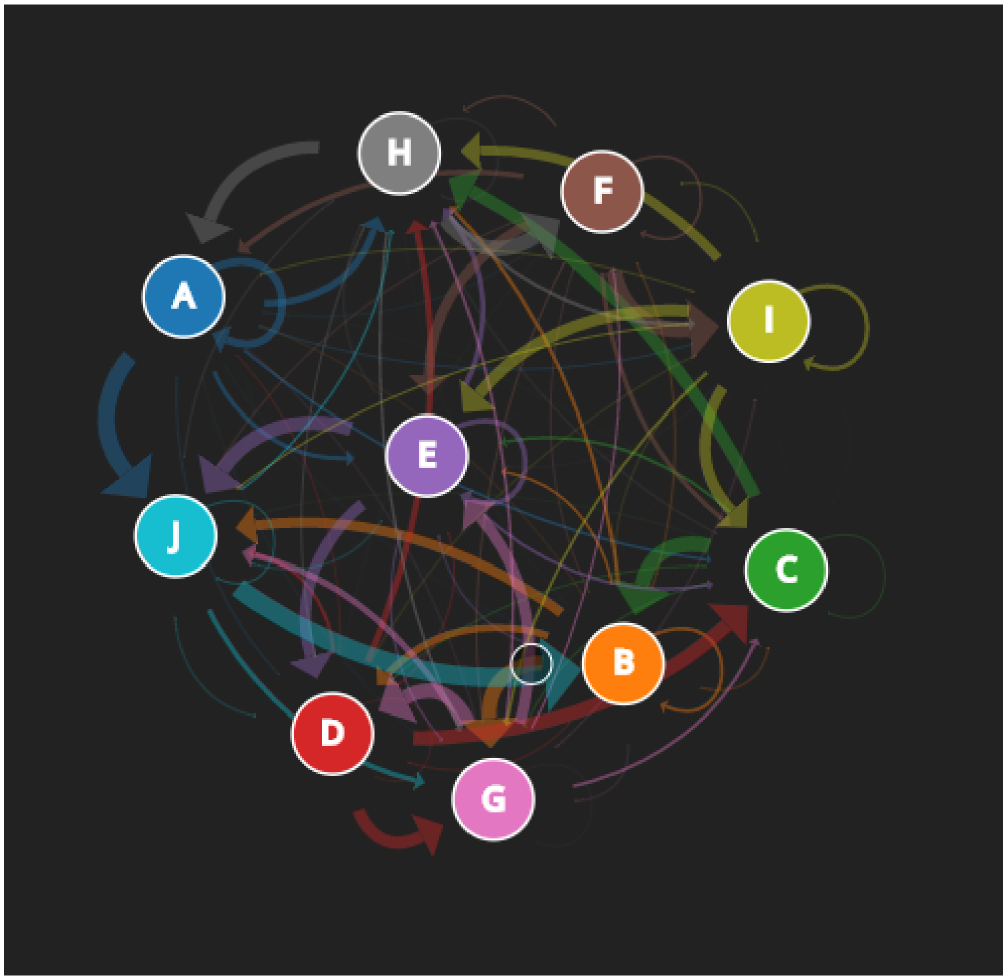
Properties of Hidden States

From looking at the most common words for each state, we can extract some part of speech meaning:

- state 0/A: conjunctions & prepositions, simile making words, who how which where when why
- state 1/B: parts of the body
- state 2/C: possessive pronouns & adjectives
- state 3/D: object/subject pronouns, comparisons
- state 4/E: adjectives
- state 5/F: conjunctions, time comparisons
- state 6/G: continuousness (being, still) and possession
- state 7/H: concepts (art, truth, self, love) and time
- state 8/I: more comparison words
- state 9/J: possessive adjectives and pronouns

We also examined the top 20 mutisyllabic words for each state. Most two-syllable words in every state had a stressed-unstressed pattern, which mirrors the fact that this pattern is the most common stress pattern

Figure 2: Transition Matrix, 10 states 20 iters



for English two-syllable words. From these lists we got the following additional characterizations of the states:

- state 0/A: negative connotation (against, thievish, wasteful), contrasting (although, without, only)
- state 1/B: nouns, positive (beauty, treasure, glory), spiritual (mistress', spirit, body), time (minutes, summer)
- state2/C: plural words (beauty's, woman's, roses, hours), possessive (our, others'), emotional (happy, fire, truly, wretched, desire)
- state 3/D: prepositions (upon, o'er, wherein), spiritual (buried, holy), values (honour, reason, respect)
- state 4/E: prepositions (upon, before, again, above), spiritual (heaven, nature, believe)
- state 5/F: nouns/nature's trials (water, winter, elements) somewhat sad adjectives (tired, absent, vainly, oppressed, alas)
- state 6/G: loss (alone, decay, away, father, morrow, heaven)

- state 7/H: beautiful adjectives (beauty, beauteous, lovely, flowers, fairest, heavenly), physical attributes (wrinkles, fingers)
- state 8/I: sounds (music), direction of feeling (against, attending, mutual, towards), bad thoughts (pity, despised) and good thoughts (fairer, praises, happy)
- state 9/J: possessive adjectives (summer's, beauty's, heaven's, precious, beloved), obscured from view (seeming, painting, inward)

Regarding transitions (Figure 2), it makes sense that state J, which is very possessive, would transition to state B, which contains a lot of nouns. Similar inferences can be made but are more tenuous. Additional data can be found in the hidden_model.txt file in our codebase.

5 Additional Improvements

Rhyme

We incorporated rhyme by randomly picking seven pairs of rhymes, arranging them in the correct order, and using them at the end of the line. The lines generated forward would often not make sense with forcing the rhymes at the end, so we switched to generating each line backwards from its ending rhyme.

Additional texts: *Hamilton* songs

We also trained our sonnets on the first three songs of *Hamilton*. These songs had a much larger variety of punctuation, in both symbols used and placement (for example, many lines ended with a dash); so in addition to removing periods, commas, colons, and parentheses, we also removed exclamation points, question marks, double and single quotes, em dashes, and ellipses. This resulted in some fairly strange grammar, particularly when it came to removing single quotes in contractions (producing words like "n" and "evry", but also unclear grammar in words like "its"/"it's"); but we still chose to remove single quotes because of the number of single-quoted quotations.

The songs in Hamilton also had many lines which were *much* shorter than ten syllables, which may have negatively contributed to the sense of the resulting poem.

1776

Throwing america some up not what,
Afar take a it what do you with split,
Not were im but im i dead gonna but,
Gonna new name face shes good out of it,
Two yo sugar to they a guarantee,
To run i its not but his to am gods,
Cousin are he ready the he squat oui,
Came and time when anybody down odds,
See socially and burr the you forgot,
Every york a get or to plenty,
Yo im dying but not so so its got,
A whoa a n up yo and i many,
Im hey but our aint until was do,
Dying down the is to hey x i brew.

Figure 3: The above sonnet, which we submitted on Piazza along with our Shakespearean sonnet, was trained using 20 states over 100 iterations. It makes much less sense than the Shakespearean one.

6 Conclusion

Division of work

Shari worked on preprocessing, getting the initial 14-line sonnet, ensuring that lines were 10 syllables with resampling, and incorporating additional source material.

Carly worked on rhyming, seeded and backwards generations of emissions, and interpretation of hidden states.

Discoveries

What are your conclusions/observations about the models you used and the sonnets generated?

- We conclude that this model captures the complexity of Shakespearean sonnets fairly well. With enough training data, we produced poems which seemed quite similar to Shakespeare. We were quite happy with the quality of our generated poems.
- On the other hand, a sonnet is long enough that our generated poems didn't stay on one topic from start to finish. This could be because our HMM only takes into account the likelihood of transitioning between two states, without looking at any prior information, and could definitely be improved upon by using more than one layer of hidden states.

Challenges

- Trying to get the poems to make sense was very difficult. For the initial sonnets, there was a lot of very poor grammar. We mostly fixed this by running the training algorithm on more data, with

SONNET SPOOFER
by Shari Kuroyama and Carly Robison

more states, and more iterations. Relatedly, incorporating different source material was harder than expected, mostly because the structure of *Hamilton* songs is so different from that of Shakespeare's sonnets. We had to make a lot of edits to the preprocessing code to get it to read the songs, and still ended up with poems which made less sense than the Shakespearean ones.

- We also surprisingly found ourselves making a good number of edits to the `HMM.py` code from Homework 5, to get the functions to output in different formats, and to generated seeded emissions. For seeding emissions, given a word we had to figure out which state that word had come from in order to produce some most likely next words. In order to do that, we chose to use the probability distribution of state transitions, but could have used the most likely state, or even kept track of the actual state throughout.
- Finally, when trying to visualize and interpret the hidden states, we found a lot of the most-used words in each state had a huge overlap from state to state. To solve this, we tried separating out the top unique words in each state. However, we still had trouble separating the states into parts of speech, as many states seemed to be separated by topic (e.g. loneliness, or flowery language) than by function.

Concluding Remarks

This project was a lot of fun. We really enjoyed playing around with the different poems, and reading them aloud to everyone who passed by.

Our code is available here.