Machine Learning & Data Mining
Miniproject 3: Shakespearebot 5000

Caltech CS/CNS/EE 155
Released March 1st, 2024

---

1 Pre-Processing

- Colab Notebook Link: https://colab.research.google.com/drive/1Yu_gH80c3yAe29Kz-LfIbe5R0f4FK7FY
- We preprocessed our data using Python's NLTK library.
- Syllable dictionary pre-processing:
  - We created a hashmap that maps dictionary words to their corresponding number of syllables by parsing syllable_dict.txt
  - A standard example like "wrinkle 2" would be added to the dictionary as {wrinkle : 2}
  - We treated hyphenated words as single words (eg all-eating would be one word) so that we could preserve their syllable counts
  - In the case of a word having multiple syllable counts, for simplicity's sake, we used solely the higher number. For example, "being 1 2" would be added to our dictionary as {being : 2}
  - In the case of a word having a different number of symbols at the end of a line, we used solely the number for when it didn't appear at the end of the line. This is because we figured the word would have a higher frequency of appearing in the middle of a line, rather than at the end. For example, "devil E1 2" would be added to our dictionary as {devil : 2}.
- Shakespeare text preprocessing
  - We tokenized the dataset to consist of words. We did not tokenize any words as bigrams, as we would need to add the individual syllable counts of words together. Since we wanted to keep the data as granular as possible, we retained each individual word's syllable count from the syllable dictionary.
  - We stripped whitespaces between lines, stripped whitespaces at the beginning of lines so that the couplets were no longer indented, removed numbers, and lowercased all words. We converted all words to lowercase because we wanted pairs of words like "From" and "from" to have the same semantic meaning.
  - For punctuation, we removed all punctuation like commas, periods, and colons (with the only exception being apostrophes and hyphens). We decided to remove punctuation at the end of lines (like colons) because we didn't want to differentiate between words such as "memory:" and "memory". Since they have the same semantic meaning, we don't think that whether a word appears right before punctuation is important information for our model.
  - For apostrophes, we thought about removing them and grouping words with apostrophes together, for example converting 'scaped to escaped so that they are the same. However, we decided to keep them separate since they do not have the same syllable counts.
  - To verify that all of our words were valid, we checked that they were in our syllable dictionary before adding them to our preprocessed dataset

- Preprocessing for HMM input
  - For our HMM input, we used a similar approach to problem set 6. We created the input X (training data) to be an array of arrays. Each array corresponds to a line in a sonnet, and they are ordered so that the first array added to X corresponds to the first line in our cleaned Shakespeare data, and so on. Each entry is an array is an integer representing its numeric identifier, which is determined by the observation mapping from words to integer.
- Preprocessing for RNN input
  - We tokenized the data by characters
  - We first separated our sonnet corpus into subsequences of fixed length, picking only sequences starting every n-th character
  - We initially chose the fixed length to be 40, as specified in the set, and n to be 20, as advised by the TAs at office hours, but we later experimented with different values of n to see which value minimized the loss
  - Using each of these fixed length sequences, we created our input and target sequences
  - For a single sequence of training data, the "input" sequence contains the first (sequence_length - 1) characters from the training sequence and the "output" sequence contains the last (sequence_length - 1) characters from the training sequence. This is because we are training the network to predict each subsequent character from the given context of previous characters.
  - Now we can use these sequences to create input and output tensors with the following dimensions:
    * The first dimension indicates the batch size, which is the number of original fixed-length sequences that make up the sonnet corpus
    * The second dimension is the number of characters in the sequence. In both the input and target tensors, this is (sequence_length - 1)
    * The third dimension is the one-hot vector representation of each character in the sequence which has a length of our alphabet, aka the number of unique characters in the sonnet corpus

2 Unsupervised Learning

We used our set 6 HMM code, since we felt like we would understand it better compared to using a package and because it was recommended in the guide to use this implementation.

Recall from set 6 that we tested 1, 2, 4, 10, and 16 hidden states. However, the sentence with one hidden state was unintelligible and it got better and better as we increased the number of hidden states. Thus, we decided to try 4, 8, 10, 12, and 16 hidden states on this project. We compared their emissions and noticed that 16 states produced the sample sentence that was most intelligible. We didn't want to increase the number of hidden states by too much though, as we did not want the model to overfit, so we did not try numbers above 16.

3 Poetry Generation, Part 1: Hidden Markov Models

Algorithm

We used the Baum-Welch algorithm from our unsupervised learning section for our HMM. When generating our sonnets of 10 syllables, each time we generated a line, we checked the number of syllables it had. If it went over, e.g. having 11 syllables, we would regenerate until we had exactly 10. We did this for all 14 lines in our sonnets.

Example Sonnet

This poem was generated by our HMM with 16 hidden states:
supposed such of now so slave doth a
appetite am from that do add reckoning
seem self and like water still more cruel
seeing you reason eye your her thy fair
unused once time's into knows not thine
honey smell of admit men's why moods my
with some glazed mine rude thought to that though
dull it chide your pleasure night and lives thou
sin of thy comfort who of to give and
own love's with feast those rose dateless to that
joy shame hath compare when faults vows happy
in were thou kingdoms that your hasten the
mind but the coward augurs with still my
for strength of crime shadows thee torn love no
Posted on Piazza. Link: https://piazza.com/class/lprgcbeelor41p/post/530

Analysis

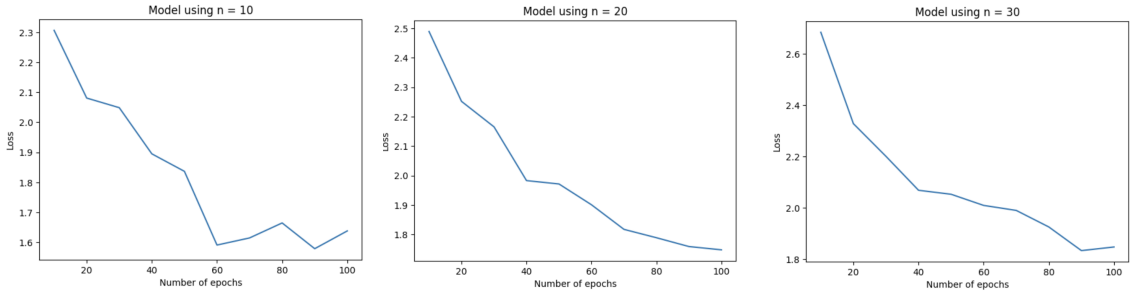
- Overall: The quality of this poem, although better than using less hidden states, is still not very high. This poem does not make that much sense as a whole, but there are certain phrases within it that make sense such as "and like water still". Compared to our HMMs with less hidden states, this is still an improvement as those sonnets have no correlation whatsoever that leads to phrases.
- Rhyme: Recall that the Shakespearean sonnets have the rhyme scheme: *abab cdcd efef gg*. Our model did not pick up on the rhyme scheme at all, and although there are some slant rhymes, it failed to capture the entire rhyming scheme. We believe this is the case because the model generated one line at a time, and because the training data did not contain information about the relationship between lines (i.e. there were no distinctions between sonnets or grouping between lines). Our HMM did not learn information about how the lines fit together, so it would not care about the rhyming of the last syllable.

- Rhythm: Recall that Shakespearean sonnets follow an iambic pentameter which is a pattern of unstressed and stressed syllables. Lines 4, 10, 11, and 13 all follow the exact meter while all of the other lines are off by one or two. We believe that our algorithm was able to pick up on this because meter is self contained within each line, and our model generated one line at a time and trained on lines of words.
- Syllable Count: The syllable count is almost perfect, with 12/14 lines having 10 syllables when read aloud. Just line 3 sounds like it has 11 syllables and line 7 sounds like it has 9 syllables. This is likely a product of only using the highest number within the syllable dictionary or not accounting for differences when words appear at the end of a line.

4 Poetry Generation, Part 2: Recurrent Neural Networks

Model Implementation

We designed our model to have a one-layered LSTM layer followed by a Linear layer, dropout layer and softmax layer. We modeled the structure of our RNN based off of the structure in the tutorial we were given. In our training we used Adam as our optimizer and calculated our loss using Cross Entropy Loss. We implemented our model using PyTorch. We tuned the hyper parameter n. We chose this to see whether varying n, and picking only the sequences starting every n-th character would increase or decrease the loss.



The graphs show that loss is minimized when $n = 10$, compared to 20 or 30. This is expected since reducing n increases the number of possible subsequences that can be extracted from the corpus, giving the model more training examples. This can lead to better generalization as the model has more context variations to learn from. However, the tradeoff is that large amounts of training examples are computationally expensive. Since they are so memory-intensive, this can lead to issues since Google Colab provides a fixed amount of RAM. Nonetheless, we chose to train with $n = 10$. If we had chosen one of the higher numbers, our model would learn less, and $n = 10$ is still not too small to the point where we would run out of memory. To ensure that our model is not overfitting when $n = 10$, we could perform validation testing, however the guide says this is not necessary.

Emission Generation: Originally, we trained the model with a step size of 20. When generating text, it would generate a few unique words and then get stuck in a loop of saying "the self the self the self the self...". We hypothesized that the issue had to do with the word "self," so we changed the emission code to never predict "self". However, this did not fix the issue, and instead the model got stuck in a cycle of saying "the snow the snow the snow...". Again, we naively hypothesized that the word "the" was the problem, and changed our emission code to never predict "the". However, our model once again got stuck in a cycle of "and my self and my self and my self and my self."

We quickly realized that our model was trained incorrectly, and decided to try lowering the step size from 20 to 10 (this was determined to result in the lowest loss as describe above). We also changed the "window" of the sequence that the model looked at when generating a prediction. Before, when predicting the next character in the sequence, the model was given the entire sequence that had been generated so far. We decided to reduce the "window length" to 40, leading to slightly more intelligible text:

```
Temperature = 0.25
weary with toil i haste me to my bed
the world that thou art as the proud love is strange
and then the eyes thee that thou art the world to thee
then thou art thee thou art the starn
and therefore when thou art thee thou art thee thou art the stere and thee the earth thence thou art all the world with mine eyes
when i am sometime day the will of all true
and so should the beauty so suches me that thou mayst thou art
and then the self to thee the world to thee
the world to make the world to be to the will
to the tring and then should be thee is best in the starn
and therefore when thou art thee thou art thee thou art the stere and thee the earth thence thou art all the world with mine eyes
when i am sometime day the will of all true
and so should the beauty so suches me that thou mayst thou
Temperature = 0.75
weary with toil i haste me to my bed
the world that thou art as the proud love is strange
and then the eyes thee that thou art the world to thee
then thou art thee thou art the starn
and therefore when thou art thee thou art thee thou art the stere and thee the earth thence thou art all the world with mine eyes
when i am sometime day the will of all true
and so should the beauty so suches me that thou mayst thou art
and then the self to thee the world to thee
the world to make the world to be to the will
to the tring and then should be thee is best in the starn
and therefore when thou art thee thou art thee thou art the stere and thee the earth thence thou art all the world with mine eyes
when i am sometime day the will of all true
and so should the beauty so suches me that thou mayst thou
Temperature = 1
weary with toil i haste me to my bed
the world that thou art as the proud love is strange
and then the eyes thee that thou art the world to thee
then thou art thee thou art the starn
and therefore when thou art thee thou art thee thou art the stere and thee the earth thence thou art all the world with mine eyes
when i am sometime day the will of all true
and so should the beauty so suches me that thou mayst thou art
and then the self to thee the world to thee
the world to make the world to be to the will
to the tring and then should be thee is best in the starn
and therefore when thou art thee thou art thee thou art the stere and thee the earth thence thou art all the world with mine eyes
when i am sometime day the will of all true
and so should the beauty so suches me that thou mayst thou
Temperature = 1.5
weary with toil i haste me to my bed
the world that thou art as the proud love is strange
and then the eyes thee that thou art the world to thee
then thou art thee thou art the starn
and therefore when thou art thee thou art thee thou art the stere and thee the earth thence thou art all the world with mine eyes
when i am sometime day the will of all true
and so should the beauty so suches me that thou mayst thou art
and then the self to thee the world to thee
the world to make the world to be to the will
to the tring and then should be thee is best in the starn
and therefore when thou art thee thou art thee thou art the stere and thee the earth thence thou art all the world with mine eyes
when i am sometime day the will of all true
and so should the beauty so suches me that thou mayst thou
```

As shown above, there is still some repetition, but it is not as severe. This could indicate that our model is overfitting to the Shakespeare text. It also seems like the temperature parameter did not affect the predictions much, also likely signifying overfitting. Also notice that the RNN did not capture the sonnet rhyme scheme. This is likely because it generates one character at a time, and generating a word that rhymes involves generating multiple characters.

5 Additional Goal - Rhyme

One issue with our original HMM sonnets was that they did not adhere to the rhyme scheme: *abab cdcd efef gg*. We started by building a rhyming dictionary, mapping a given word to a word that rhymed with it. We built the dictionary by iterating over the last word of every line of the input data. After the dictionary was built, to generate a pair of rhyming sentences, we first chose a random key from the rhyme dictionary. We seeded that word at the end of the emission and a random state was chosen to represent that emission. Then we examined the column of self.A that corresponded with that state to find the previous state that had the highest probability of leading to our current state. After choosing the highest likelihood state, an emission was generated using self.O.

This method was successful in making the sonnet follow the correct rhyme scheme, but did not make it more intelligible in other aspects.

The image below shows our original HMM sonnet with 16 hidden states and NO rhyme scheme implementation.

```
that out determinate heats more as what
inward draw old happier are in that wronk
bear more glorious up dressing confess look
your defects think from old bewailed like
a thou become one year where-through you the
bettered so she tender be and i is
evermore say by be thee short till age
is and bounteous of naming the pleasure
themselves wish survive thee thy signs never
but lived reproving robe the brain of still
my worth of the account heaven was thy
i you straight thy waste proud untrimmed let to
until weeks governs be then like profaned
i shine when the thou quick your his blood your
```

The image below shows the sonnet generated by our HMM with 16 hidden states WITH the rhyme scheme implementation.

```
broils since seems no for what and bareness state
when gardens shall excuse why in misplaced
date there out these groan lascivious he fate
vex he with the crowned while i disgraced
not beseechers whose tells with define rare
o that thanks will mournful best the the skill
or sweets and but compared having when are
eye nor do in me eye a fast doth still
so i not an already love perish
o o mark but o o o for and cheek
art give made esteemed think is all cherish
for and seems but but angry mark yet seek
then were call almost thou second awake
and o and he o lest yet sweetness sake
```

As shown above, the sentences in both sonnets are relatively not intelligible. Recall that our backward state generation gives us a fixed sequence of states given the last state, whereas with forward state generation the sequence of states is not fixed given the first state. In practice, this means that with our backward scheme, there is less variety in the words, and we can see that in the sonnets above. In the sonnet generated with the rhyme scheme, the word "o" is repeated a lot. This was one of the main tradeoffs we noticed between accuracy and creativity.

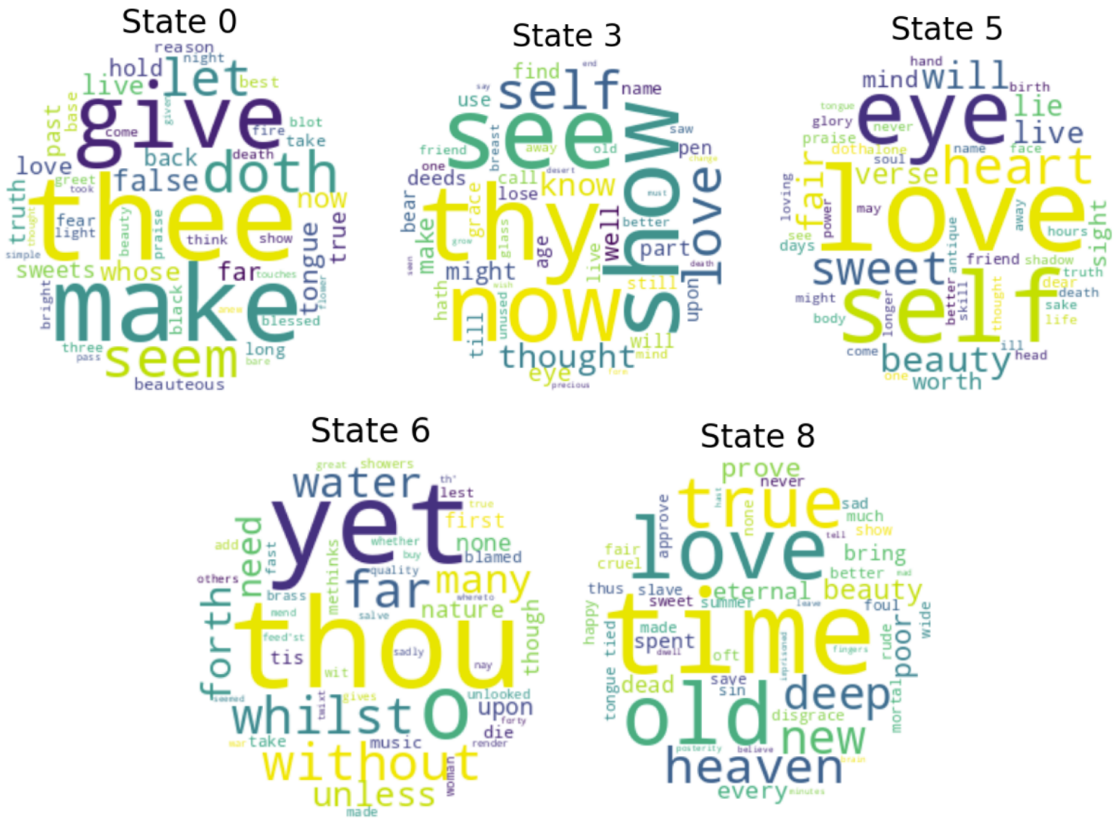
We hypothesized that a possible fix would be training the model on the backwards data. Since the model would be trained to predict sentences starting with the last word, it would be easier to seed the last word. Another fix could be to leverage properties of Markov Chains to find the probability distribution of y_n given y_{n+1}. We know the (i, j)th entry of A represents $P(y_{n+1} = j|y_n = i)$ where y_n is the state at time n. Using Bayes' rule,

$$P(y_n = i|y_{n+1} = j) = \frac{P(y_{n+1} = j|y_n) \cdot P(y_n = i)}{P(y_{n+1} = j)}.$$

We know $P(y_{n+1} = j|y_n = i)$ from the entries of our transition matrix A. However, it is not straightforward to estimate $P(y_n = i)$ or $P(y_{n+1} = j)$. To do so, we need to find the probability distribution of the state space. If we find this probability distribution, then we know the values of $P(y_n = i)$ and $P(y_{n+1} = j)$. If we assume that we can get to any state from a given state in a fixed amount of steps (A is irreducible), and notice that our state space is finite, then by properties of homogenous Markov Chains we can conclude that A admits some invariant probability, π. By the Ergodic Theorem, we know that overtime, the probability distribution of the state space will converge to π. Thus we could compute π and use it to sample from y_n when guessing which state came before y_{n+1}. However, the issue with this would be that finding π can be computationally expensive, and we would have to check that A is irreducible first.

6 Visualization & Interpretation

We used the wordcloud package, since this offered interpretable representations of our data in problem set 6. We chose to highlight the 5 wordclouds with the most discernable patterns (from the 10 possible states).



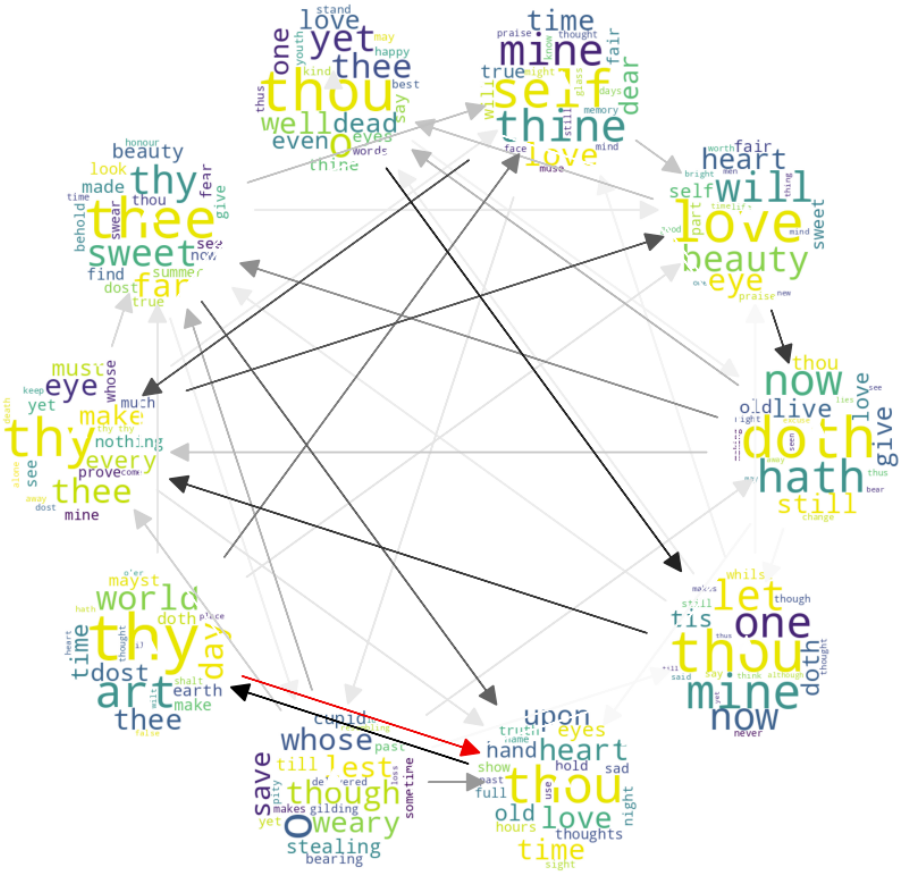
We recorded the following patterns within these states:

- *State 0*: Mostly 1-syllable words.
- *State 3*: Mostly 1-syllable words.
- *State 5*: Has to do with the body and love. We see words like ”heart”, ”eye”, and ”beauty”.
- *State 6*: Higher frequency of multi-syllable words such as ”whilst”, ”without”, and ”water”.
- *State 8*: Related to aging and time. Some of the most represented words are ”time”, ”old”, ”eternal”, and ”heaven”. A noticeable characteristic is that many of the common ”filler” words like thy, thee, and thou are not represented.

Here are the results of the top ten most popular words for each of the hidden states (0-9):

Hidden State	Top 10 Words
0	'to', 'is', 'that', 'it', 'in', 'as', 'and', 'he', 'me', 'but'
1	'and', 'that', 'me', 'you', 'thee', 'which', 'world', 'by', 'so', 'as'
2	'and', 'but', 'to', 'o', 'doth', 'so', 'for', 'then', 'even', 'that'
3	'in', 'and', 'to', 'as', 'be', 'not', 'for', 'thy', 'see', 'against'
4	'my', 'i', 'thy', 'your', 'the', 'their', 'his', 'it', 'thee', 'no'
5	'love', 'self', 'not', 'heart', 'have', 'am', 'will', 'sweet', 'own', 'do'
6	'if', 'for', 'when', 'but', 'which', 'so', 'i', 'yet', 'as', 'how'
7	'the', 'me', 'my', 'a', 'not', 'that', 'thee', 'this', 'love', 'art'
8	'the', 'to', 'a', 'that', 'and', 'time', 'this', 'being', 'in', 'love'
9	'of', 'thou', 'with', 'in', 'to', 'is', 'on', 'for', 'are', 'that'

Here is a visualization of the transitions the HMM takes between states:



Analysis and Interpretation Based on the patterns we see in states 0, 3, and 5, we can see that the HMM is able to learn some parts of Shakespeare's rhythm and meter. For instance, the states with a single syllable might correspond to a part of the text where short words create a sense of abruptness or rapidity that contributes to the poem's rhythmic qualities. Similarly, the HMM is able to learn patterns for words with multiple syllables. For instance, they might occur in common areas of sonnets where Shakespeare wants to express emotion or use precise language. Also, we observe that the HMM is able to capture some thematic features, as evidenced by hidden state 5 and hidden state 8.

Some patterns we noticed between these transitions:

- All arrows pointing to the hidden state corresponding to multi-syllable words are light-colored, signifying low transition probabilities. This makes sense because it is likely that complex words appear less frequently in the poem.
- The darkest arrow coming from the state corresponding to body parts/love is pointed at another state that with frequent occurrences of ”doth”, ”hath”, and ”love”. It makes sense that a description of someone's love and body might be followed by a verb (i.e. ”have” or ”does”) that indicates they possess those qualities.
- We notice an transition outward from the word bubble containing ”world”, ”time”, ”earth”, and ”day” to a word bubble containing words that have to do with self-reference terms, like ”mine”. This could indicate a relationship in Shakespeare's poem between self-reflection, aging, and broad ideas surrounding identity and place in the world.

This leads us to believe that the HMM is able to recognize groups of words that have semantic meaning. For instance, transitions indicate that noun and verb pairs are common (in this case, these nouns have to do with the body/love and the verbs have to do with possession). Our HMM is also able to pick up on words that are similar in meaning or theme.

7 Extra Credit

Additional goal: Spenser dataset

We aimed to incorporate the Spenser dataset so that our HMM would have more data to work with. This added new patterns for sonnets that our model could potentially pick up on. We did not add new words however. Since the syllable dictionary only includes words from Shakespeare, we removed words from the Spenser dataset that were not also in Shakespeare. Since the Spenser sonnets follow the same patterns as the Shakespearean sonnets, even though they were not written by Shakespeare himself, they could still serve as beneficial training data. The best sonnet we generated with the combined dataset was with 10 hidden states, producing the following sonnet:

with more meditation then my soul's then
since by any fire that travel of to
is power the may them and fierce with woe of
away is they happy woe they my and
confounds to call a flowers so as thou
for all mark despised will back but then
for no with his down give all days winter
so once once of thy praises to each were
thereof powerful the to life in friends where
that take with my desire canst gaze painting
i thy fault that one lively shamed not spend
heat being fire in the decay ever
thy devise a her state twice be mine not
well to to unknown the birds corrupting

Based on this output, the model with the combined dataset performed relatively similar to the HMM using just the Shakespeare dataset. The syllables are still correct, and the meter is still mostly correct, while the rhyme is incorrect. This sonnet seems to have stronger motifs though, with strongly connoted natural words like "flowers", "fire", and "birds". Since the sonnet did not improve that much, this makes us think that a lack of data is not necessarily the problem, but instead our model should try to focus on learning different aspects of the dataset to perform better.

Additional goal: Generating other poetic forms

We were curious to generate poems with different syllable schemes using the sonnets dataset. For instance, a haiku. We created code that would allow us to input any array corresponding to a syllable scheme, and output poems of that form. A haiku is generated using the array [5,7,5], since a haiku is a poem with three lines containing 5, 7, and 5 syllables per line. Here are some examples of poems we generated using an HMM with 100 iterations and 8 hidden states. We chose this number of iterations and hidden states because it led to more intelligible poems:

- Haiku:

  blunt nature's change dost

  truest woe is the self on

  truth place you the love

- Nonet (9-line poem that has 9 syllables in the first line, 8 syllables in the second line, 7 syllables in the third line, and continues to count down to one syllable in the final (ninth) line):

  in so concealed honour most my that

  use from look my heaven in guides

  that the jealous i i of

  by look beauty and his

  what things that thy thy

  slave think thy glass

  yet bear all

  where thou

  true

- Poem of our own form ([4,8,4,2,2], we just chose this for fun!):

  shadows wish a

  not thee unstained mind most these

  and let tongues me

  look large

  o when