1 Pre-Processing

Initially, we made following preprocessing choices for HMMs:

- *Ignore select punctuation such as ',', '.', '!', '?', '()'.* The idea behind this decision is that without removal of these characters are naive HMM sonnet generation algorithm would have to ensure two punctuation marks would not be emitted in a row as well as uphold the rules of parenthesis. We decided to avoid this endeavor in order to see purely how the words were modeled, as punctuation does not seem to play a particularly defining role in Shakespearean sonnets.
- *Keep hyphenated words and contractions as single word.* We made this choice because often hyphenated words are meant to be interpreted as a single word and make sense in a different context than each of their separate components. For example, in the word 'self-substantial' (sonnet 1), self modifies the meaning of substantial and should be treated as its own word.
- *Label encode each word with the end-of-line (EOL) encoded as its own word* The decision to create an encoding for the end of the line was motivated by the fact that the lined style is a very distinctive and important feature of a sonnet beyond its sentence structure. We felt it was important to encode in our model, since there may be some significant patterns in the types words that tend to precede or follow EOL's. Note that this decision was made somewhat later on as we attempted to generate sonnets and wanted to figure out a good way to know when a line had ended.
- *Interpret a stanza as a single sequence. Namely, each quatrain and each couplet form a single sequence.* This choice was driven by the fact that all rhyme-scheme patterns in sonnets are contained within a stanza rather than a line. We decided against a full poem as a singular sequence because that would significantly reduce the amount of training data we had and force the HMM to maximize the likelihood of very low probability events (very long sequences), taking away precision from the maximization step. Note that we excluded 3 sonnets that more or less then 14 lines, because we wanted to encoded stanzas as only quatrains and couplets for consistency.

Enacting these pre-processing decisions, resulted in what we felt were fairly reasonable generated sonnets, so we didn't feel the need to make any significant changes. One thing that we did decide to add in later was punctuation, just to add more complexity to our sonnets. So, we ended up tokenizing punctuation as separate words using the `nltk.tokenize.Tweettokenizer.tokenize`. We chose this tokenizer, because it abides by our decision to keep contractions and hyphenated words but separates punctuation as separate tokens.

2 Unsupervised Learning

We used the TA solution from HW6 for implementation of HMM Unsupervised training. We tried using 5, 10, and 25 hidden states all trained on 200 iterations. Note that we only trained on 77 of the 151 (number of 14 line sonnets) sonnets, which was 308 of 604 sequences/stanzas. We evaluated the performance of each model by looking at the resulting quality of generated sonnets. We found that the generated sonnets for 5,10, and 25 hidden states were somewhat similar in quality, and difficult to compare qualitatively. All retained some of Shakespeare's voice, though rhythm and rhyme were mostly lost in the strictly desired sonnet-form. There were some lines that made more sense in terms of sentence structure than others across all poems. One thing we noticed was that generating two of the same words in a row seemed to be more of a problem with the 5 state HMM compared to the other models. We ended up choosing 25 states somewhat arbitrarily as we felt that perhaps there was somewhat more logical sentence structure there. See the next section for examples of 5, 10, 25 hidden-state naively generated sonnets.

3 Poetry Generation, Part I: Hidden Markov Models

The following was our naive algorithm for generating a 14-line sonnet. Note that we enforced rhyming in a different algorithm described in the additional goals section. The inputs into our naive algorithm are the O and A matrices of the trained HMM as well as the `min_length` and `max_length` of a particular line. We included these extra length parameter because we found that it made the syllable count closer to the desired 10 syllables a line. We achieved the best results with `min_length` = 6 and `max_length` = 10.

Naive Algorithm:

```
1. Let emission denote the final sonnet. Let state be initialized uniformly at random among the
   hidden states. Let $O$ and $A$ denote the observation and transition matrices, resp., of our trained
   HMM. Repeat until emission has 14 lines:
   (a) Let line denote the current line of the sonnet being generated. Save the start state of the line as
       first_state_of_line. Repeat until an EOL observation is emitted:
       i. Sample an observation from the probability distribution defined by $O_{\texttt{state},:}$ (the row of
          the observation matrix corresponding to the current state. Add this observation to the
          line
       ii. Sample the next state from the probability distribution defined by $A_{\texttt{state},:}$. Set state as this
           newly sampled next state.
   (b) If the generated line has greater than or equal to min_length words (not including the EOL)
       and less than or equal to max_length words (not including the EOL) add this line to the
       emission
```

Consider the following examples of naively generated sonnets based on HMMs trained over 200 iterations with 5, 10, and 25 hidden states, respectively. Note that each were restricted with `min_length` = 6 and `max_length` = 10 :

5 Hidden States

```
bud thee thing though wrinkles living not some,
of of minutes pleasure , nights basest slander's stelled,
tells shall moon, there and many give,
the weight heavenly long dust the love eternal life,
hideous painted advised are anon more
on doth in reign? to thou youthful their,
of though not if tincture millions down sheaves,
eyes copy by if be love to none
for defendant renewest why my truth live upon
but by space thoughts beard,
still graciously till you hath love in store,
were is deeds to must am
as wake by be to their thine by
and your your, bones are gentle before unperfect
```

10 Hidden States

```
other? all were though my will loving eyes ruinate
to the far i when nothing happy this,
did day of my so winter her,
till subject disgrace not misplaced,
the use were feel his himself temperate,
neither unseen those so self,
pleasure do be perpetual single glorious
the celestial pay roses end may it other,
full public with thou the set,
thou our to woe should father.
crawls i the known made,
doth unhappily yet time your that,
the death-bed, where mine honour,
and cheek my it the your like cannot?
```

25 Hidden States

```
day's on deceased age, which view thy sight,
which blessed of from deny:
and by not reign with where after art,
that i the shown,
or first-born to bright truth it tillage love review,
lost one , before vanished unthrifts.
by wrought the lose no do my young,
and live his worth pen
each thing told bare it behold,
with far fore-bemoaned air,
spending for a rich in for with be,
is praising bear of self.
engraft entertain those of?
make thou with audit blood
```

As mentioned in the previous section, the qualitative difference in generated sonnets between different numbers of hidden states was very small and not particularly striking. Across all numbers of hidden states, Shakespeare's voice was somewhat present, in the sense that the most commonly used words in Shakespeare tended to appear in the generated sonnets. The HMMs were able to capture Shakespeare's vocabulary as the generated poems only sampled words from a support set that only consisted of words in the Shakespeare data set. Moreover, since the HMM is a probabilistic model that seeks to maximize the likelihood of the training data, it would make sense the words that appear with high frequency in Shakespeare's sonnets would have a high probability in the resulting observation matrix of the trained HMMs. Some examples of the more probable words include "love", "self", "thee", "thou", "beauty", "time", "old", "night", sweet".

Although the sentence structure is fairly muddled and not completely logical in any of the HMMs, we qualitatively assessed after generated many sonnets for 5,10,25 hidden states that on average the 25 hidden state model had the most grammatically recognizable sentence structure, in a very loose, somewhat arbitrary sense. One motivating factor in this assessment was that the 5 state HMM tended to generate sonnets that would frequently sample the same word twice in a row, while the 10 state HMM would misplace punctuation more often than the 25 state HMM. For instance, in the 25-hidden-state example above there seem to be adjectives modified nouns like "deceased age", "bright truth", "fore-bemoaned air", "my young" (possessive-adjective,noun). Moreover, there appear to be subordinate clauses preceded by "which" in this 25-state example, including "which view thy sight" and "which blessed of from deny." Another grammatical form that s somewhat present in this same sonnet is "and" joining clauses together. The HMM may have been able to capture this grammatical pattern in Shakespeare's sonnets, by having high transition probabilities from states that output adjectives with higher probability to states that output nouns with higher probability. Similarly, using "which" and "and" as a joining word may have been realized by the HMM by transition to a state that outputs "which" or "and" with high probability after being in a state that often outputs an end-of-line (EOL) character or punctuation. Speaking of punctuation, the HMM did a very good job of recognizing that punctuation is often placed at the end of a line in Shakespeare. The model was likely able to do so through a high transition probability between a state associated with punctuation and a state associated with the EOL.

Compared to what a sonnet should be, the alternating rhyme scheme, iambic pentameter rhythm, and 10-syllable lines are not really upheld in the naively generated poems. However, in terms of the 10-syllable lines, the HMM gets somewhat close to having lines that are on average 10 ± 2 syllables, which is also helped by having the `min_length` and `max_length` parameters. Additionally, the poems do not really make any sense apart from the retention of the most common words used by Shakespeare (e.g. "love", "death", "time", etc.), which give the poem a vague notion of subject.

4 Poetry Generation, Part II: HMMs with Rhyming

After noting that the naive sonnets did not maintain the rhyme scheme expected of a Shakespearean sonnet, we explored creating a rhyming dictionary to seed the last words of each line in our poem generation in order to enforce the rhyming. In order to accomplish the generation of each line backwards, we trained an HMM on stanza sequences that were reversed. In this fashion, the learned forward transitions between states would now correspond to generating a line backwards. For the following explanation of the algorithm we will denotes Y_i as the state at step i and X_i as the observation at step i.

In terms of choosing the start state, we wanted to sample from $P(Y_0|X_0 = \text{EOL})$ as we know we are starting from the end of the line. We also made the assumption that the distribution $P(Y_0)$ is uniform. Then, we know that $P(X_0 = \text{EOL}|Y_0)$ is stored in the column corresponding to 'EOL' in the observation matrix O, or $O_{:,\text{EOL}}$. Since we have to multiply each of the probabilities in the vector $O_{:,\text{EOL}}$ by the uniform density and then normalize the resulting vector to get $P(Y_0|X_0 = \text{EOL})$, we know that due to the uniformity of $P(Y_0)$ we can just normalize $O_{:,\text{EOL}}$ to get the same result. Additionally, we have pre-chosen the last word of each line so we would like to sample our second state Y_1 for each line from the distribution $P(Y_1|Y_0, X_1)$. This distribution can be calculated as follows:

$$\frac{P(X_1, Y_1|Y_0)}{P(X_1|Y_0)} = \frac{P(Y_1|Y_0)P(X_1|Y_1)}{P(X_1|Y_0)}$$

Next, we can rewrite this expression using A and O. Note that the vector multiplication used below is element-wise multiplication.

$$= \frac{A_{Y_0,:} \odot O_{:,X_1}}{\sum_{y_1} A_{Y_0,y_1} * O_{y_1,X_1}}$$

Rhyming Algorithm:

1. Let `emission` denote the final sonnet. Let `state` be initialized according to probability vector $O_{:,\text{EOL}}/\sum_y O_{y,\text{EOL}}$.
2. Let O and A denote the observation and transition matrices, resp., of our trained HMM.
3. Choose random line endings that satisfying the rhyme scheme
4. Repeat until `emission` has 14 lines:
    (a) Let `line` denote the current line of the sonnet being generated. Save the start state of the line as `first_state_of_line`.
    (b) Let `line_ending_word` denote the last word chosen for this line
    (c) Let `p` $= \frac{A_{Y_0,:} \odot O_{:,X_1}}{\sum_{y_1} A_{Y_0,y_1} * O_{y_1,X_1}}$
    (d) Sample the state associated with the last word, `state`, from the probability distribution defined by `p`
    (e) Repeat until an EOL `observation` is emitted:

   i. Sample an `observation` from the probability distribution defined by $O_{\texttt{state},:}$ (the row of the observation matrix corresponding to the current state. Add this `observation` to the `line`
   ii. Sample the next state from the probability distribution defined by $A_{\texttt{state},:}$. Set `state` as this newly sampled next state.

  (f) If the generated `line` has greater than or equal to `min_length` words (not including the EOL) and less than or equal to `max_length` words (not including the EOL) add this `line` to the `emission`

5. Once `emission` is finished reverse each of its lines to get the final sonnet

An example of a poem generated by this method with an HMM trained with 25 states using 200 iterations is as follows (note that this poem was created before punctuation marks were included as tokens):

```
and but life give tender spring
upon than thy art maiden grow
or gave the time bud canst but thy hath sing
they through but those thou were ten self a go
to stage drawn much mistress fair thine others frame
to it in endowed the cold as rhyme
huge it woman's age behold will many my same
o most fond child the say for and all-eating time
of issueless wail outward no fairest shines
if who is confounds make song
thou each thy or thriftless allow sweet declines
look thou thine thy look lusty find a wrong
a still eye wind may nought your scythe world days
against shake tells hath world face some praise
```

Evaluating the sonnets generated using this algorithm qualitatively against the sonnets generated with the naive, we can notice a slight trade-off in the creativity of the sonnet since we are constraining the poem generation through the last words of each line. However, there was no significant decline in the quality with regards to sentence structure.

5 Visualization and Interpretation

As elaborated upon in the previous section (4) on poetry generation, the HMM likely learn patterns in Shakespeare's sonnets by maximizing the likelihood of trends between words seen in the training sonnets. Namely, the HMM seems to associate certain states with a "type" of word whether that be an adjective, noun, punctuation, EOL, or conjunction and having higher transition probabilities between states that correspond to some grammatical patterns. Some such patterns mentioned in that previous section include adjective modifying nouns, conjunctions appearing at the start of the line, and punctuation appearing at the end of a line. Moreover, HMMs skew towards words that often occur in the training data, learning a high probability of observing these high-frequency words, often having a high probability of observing these words given several different states.

As we continue to analyze the HMM in this section, note that the primary methods used in our analysis were qualitative assessments of the poem generation discussed above combined with the most probable state-to-words table and transition diagram discussed below.

For ease of interpretation, we will visualize a 5-hidden-state HMM below, since it did not have any significant qualitative differences in its generated sonnets compared to the 10- and 25-hidden-state models. In the table below, we have the top 10 most probable words (top=most to bottom) given a certain state:

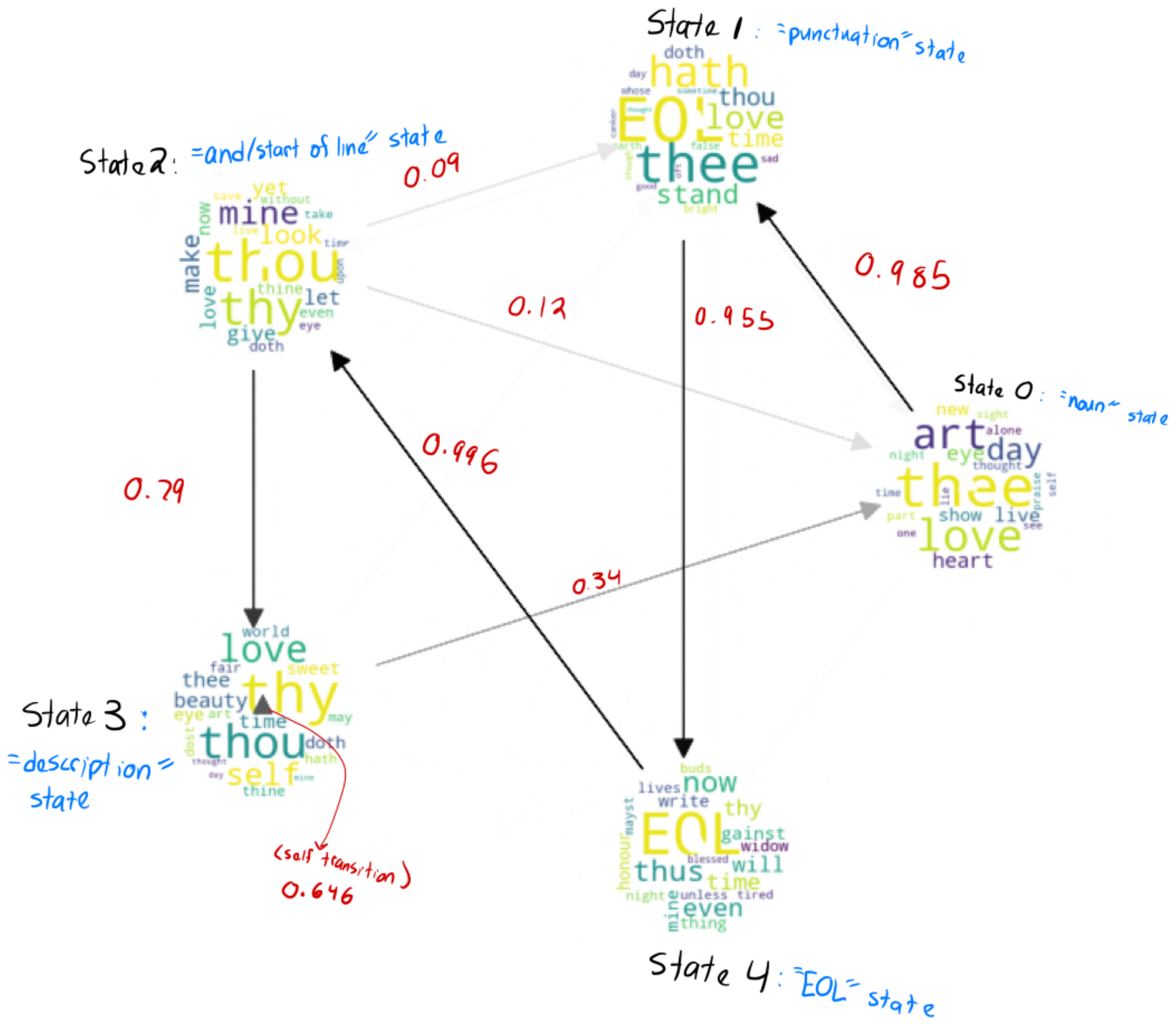
State 0	State 1	State 2	State 3	State 4
'thee'	','	'and'	'of'	'EOL'
'me'	'.'	'the'	'the'	'and'
'the'	':'	'that'	'my'	'to'
'heart'	'?'	'to'	'in'	'o'
'love'	')'	'thou'	'thy'	'or'
'this'	'a'	'for'	'i'	'that'
'art'	';'	'when'	'to'	'but'
'be'	'EOL'	'with'	'thou'	'than'
'live'	'so'	'i'	'that'	'in'
'new'	'not'	'of'	'with'	'when'

The following table represents the corresponding (approx.) probabilities of each word given the state to which they are associated :

State 0	State 1	State 2	State 3	State 4
0.02	0.5	0.08	0.0377	0.6
0.017	0.1	0.04	0.0367	0.04
0.011	0.04	0.03074	0.03	0.02
0.0098	0.02	0.03071	0.027	0.014
0.0091	0.014	0.027	0.026	0.0124
0.00893	0.0116	0.025	0.024	0.0122
0.008872	0.01	0.023	0.023	0.011
0.0082	0.007	0.022	0.01734	0.0099
0.0069	0.00634	0.021	0.01732	0.0095
0.00685	0.00632	0.0209	0.016	0.0094

For state 0, we see that the top 8 words that associate with the state are are all nouns (with 'me' and 'thee' being objective pronouns) except for 'the'. For state 1, we see that most of the top 10 words are either types of punctuation or the EOL line character. For state 2, we see several of the top 10 words are prepositions such as 'to', 'with', 'for', 'of'. However, for state 2, 'and' is by far the most probable words being at least twice as probable as any other the other words in the list. Due to these two observations, state 2 might be a state representing words that follow or precede nouns. For state 3, we see that both 'my' and 'thy' are present with are both possessive adjectives. In this state, we also see some associated pronouns like 'i' and 'thou', as well as prepositions like 'in', 'to','with', and 'of'. Finally, for state 4, it seems that the state is (by far) mostly associated with 'EOL' the end of line, but also corresponds to words that might occur at the very beginning of a line i.e. joining clauses after a line has just ended.

We can use these interpretations of what groups of words associated with a state have in common together with a visualization of the transitions within state to further analyze the workings of the HMM. Note that in the following visualization word clouds generated from a particular state are used to denote that state in the Markov Chain graph, but we found that these word clouds were much less useful in analyzing what these states represent. This is because the word cloud library used excluded punctuation and common words line 'and', 'the', 'me', etc. that is dubbed as 'non-meaningful'.



Note that in the figure above the transition probabilities are labeled in red and a descriptive name is given to each state based on our observations about the top ten words associated with each state. Based on this analysis, the HMM can be interpreted to work as follows: start of line state likely transitions to a description state like 'thy', which either transitions to itself and generations another 'description' word, or transitions to the noun state. Then, the noun state is followed by the punctuation state which then likely transitions to the end-of-line states. After the end of line state , we return to the start-of-line state and with significant likelihood generate 'and' to signal a new clause for this new line.