1 Introduction

- **Group members**
  Spenser Schneider Matt Riker
- **Team name**
  Jawns
- **Division of labour**
  Matt: Pre-Processing, RNN, Unsupervised Learning, Report Spencer: Pre-Processing, HMM, Unsupervised Learning, Visualization, Report

2 Pre-Processing

For pre-processing the data for the unsupervised model and the HMM, we decided to split up the data by line. We decided to make all of the letters lowercase when pre-processing, and we also took all of the punctuation out other than the apostrophes and hyphens. We decided to do this because we did not want words that contain apostrophes and/or hyphens, such as ”ne’er-cloying”, to turn into the word ”neercloying” which is just gibberish. Furthermore removing the apostrophes would’ve changed the meaning of words whether is be from plural to singular or just a different word altogether. It is important to note that during post-processing, or after out sonnets were produced, we converted any single ”i” to an ”I”. We performed this after so that we would not have capitalized ”I” in the middle of a word. We tokenized the dataset by going through and assigning each word a distinct number. If that word was seen again in a different line or stanza if was reprented by the same number that was previously assigned to it. One thing that changed throughout the project is that in the LSTM each distinct character was represented by a distinct number whereas in the unsupervised model and the HMM each distinct word was represented by a distinct number. Furthermore, in the LSTM model, we kept all punctuation characters and assigned them to a distinct value as well, opposed to only keeping the hyphens and apostrophes in the HMM and the unsupervised model parts. Another difference that had to be accounted for with the different models is that we stripped the newline, or ” n” characters in the Unsupervised and HMM models and hard-coded this in for a new line to be formed after the correct number of syllables. On the contrary, we left these characters in for the Recurrent Neural Network as we allowed for a new line in the poem to be formed each time the model predicted a new line character was the next character.

3 Unsupervised Learning

We used the HW6 solutions to implement the Baum-Welsch algorithm. The EM algorithm for unsupervised training of HMMs is
1. INIT: randomly initialize HMM model parameters
2. E-STEP: run Forward-Backward to compute marginal probabilities $P(y_i^j = a, \mathbf{x_i})$ and $P(y_i^j = b, y_i^{j-1} = a, \mathbf{x_i})$ based on the current model parameters
3. M-STEP: update the model parameters based on the maximium likelihood estimate given the data
4. If not converged, repeat from Step 2

We used basic python to tokenize the data and create the requisite dictionaries to run the HMM. We chose the number of hidden states by running the HMM with different numbers of hidden states and judging the readability of the sonnets. We ended up using an HMM with 10 hidden states. More than this produced repititions and gramatical errors and less than 10 produced random giberish.

4 Poetry Generation, Part 1: Hidden Markov Models

To create a 14 line sonnet we generated words until the line reached at least 10 syllables and then, if it was less than 11 syllables we added it to the sonnet. We allowed this extra syllable because Shakespeare did this himself in some cases. He would turn a 3 syllable word into a 2 syllable one at the end of lines for example. We counted the syllables using the syllable dictionary. We then put in punctuation and capitalized the first word of every line.

The rhyme and rhythm are very out of wack when compared to a true sonnet.The syllable count is true to what a sonnet should be. This is because our algorithm produces lines of 10 or 11 syllables. In general with very few hidden states the sonnet is poor because it is essentially just choosing random words from the hidden states. As the number of hidden states gets above 4 or 5 it starts making more sense until getting into the teens where the large amount of states leads to more gibberish poems. The best amount of hidden states appears to be in the 6-12 range. This is an example poem:
O this but tongue o shamed love sleep being till,
Eternal then hot subsist thou thee was,
Thine art the pleasure lov'st to beams rain now,
Admitted bearer as I when was conquered,
Every proud come mightier if that famine,
To merit added nor harder which night,
Feel or was made at on you here yellowed,
Husbandry are stand when twilight day your,
Me which defeat yet my men's finding though,
Love's them birds cruel sea lap brain none more,
Seals break and do then slandering tell you than,
Grow nothing to habit much the eyes nature,
Yet that thy blood wilful is to how come,
Action thine says that and are shall born my.

We can see some sentences have some semblance of following iambic pentameter, but is isn't consistent. The poem retains his voice in the sense that it is still Shakespearean English which makes sense because all the words are indeed his. While the HMM was able to capture some grammatical structure, as the poem isn't complete gibberish, it still doesn't make complete sense.

5 Poetry Generation, Part 2: Recurrent Neural Networks

We implemented a character based LSTM model, with a single layer of 200 units, and we used a fully-connected output layer with a softmax nonlinearity, and categorical cross-entropy. We used a training set of 40 characters and used semi-redundant sequences, picking each starting at the 5th character. We used keras for the packages, and from keras used things such as a sequential model, a dense layer, and the dropout. First, we prepared the data, by going through the text and finding all possible text characters (including symbols) and mapping them to a specific number and vice versa using two dictionaries. We then split up the data using a sequence length of 40, and only picked sequences starting with every 5th character to speed up the training process. Also, we hot encoded our data for the model to train on. We trained the model using the described parameters below. Once we trained our model, we compiled it and then produced a poem by predicting the next character one at a time based on the previous 40 characters in the poem. Since our poems produced many obscure words and sentences it is tough to tell exactly which temperature produced the most accurate sonnet. However, we can see that the poem produced with a temperature of 1.5 produced some words and phrases that made little to no sense at all. For example, from that poem the sequence of words: ”cntnsi,popi,booooony oooof” was produced. This makes sense since we know that temperature is used to control the randomness of the predictions, thus, the model computes the softmax divided by the temperature. Thus, when we increase the temperature the softmax is being performed on smaller values, which leads to a less certain probability and a greater diversity in the predictions. So, when we divide by high temperatures, such as 1.5, we can expect more random predictions, and expect a less accurate poem prediction. However, to reiterate, this was difficult to determine by simply looking at our poems due to the inaccuracy that our model produced. Furthermore, we used ModelCheckpoint from keras to help store the file path of our model that performed the best. Some of the parameters we changed were batch size, number of epochs, and dropout. The model that produced the highest accuracy and lowest loss was a dropout of 0.2, with a single layer of 200, 400 epochs, and a batch size of 50. These parameters lead to each epoch running for approximately 45 seconds We used a sequential model with a single layer of 200 units, and Adam as the optimizer, with the metrics of accuracy. The poems that my model produced were not entirely accurate. Many of the words in the poems were not actual shakespeare words, however, there were some words sprinkled in that are a part of the shakespeare dictionary. No, the LSTM does not exactly learn sentence structure or sonnet structure either. The HMM quality of the poems is much better than the poems produced by this LSTM, and this makes sense since the LSTM is based off of each character, which leads to more words being produced that dont much sense at all.

Temperature: 0.25:
shall i compare thee to a summer’s day? fn nent fear amt mifa birered mnte the feasty oake, and ieavu shou hnstny noga earh a pave, lnt tie hias h ho motuenled blone blthruaysed bmd desel ofe munpllg, she srises that the worth mu oind cy wouth gy sork shy sorrt, mow ttchdhed hsrh mi toietest’s bre she world, oot hreu the heas hn beauty tp mi wourt, doth botme bld shemes and do bll dhg heeo. oo ohne an mnt ce wrreuesy mr oueh roond bw ary iord inve) which mu tiis worrh aitirtare tomeu, dooh rensin aelot dot lovh whll oyes gesl: to dause of su me anth mad

Temperature: 0.75:
shall i compare thee to a summer's day? fn iint wiln tereennnss woutu, gor mit statd to brny rfpugd art, to mycu- the iintire wou telde thee whth meeht toind, oos hrrbnny siou art foauted teseumint, anl thy bue mo linc cyt tiat lo oe ayr domf, bnd fild ou marelred thyhnl roevertiog eyes: wiich havh ho ny hrabi in hir cecrgog, that io thy bueuna no clournnnsedn's orot: then teof aelpoe thy beauty shy ceckgs, shen dot anr my blsotye mfw ceauty shy simuld liv. fo no, oore what is isse theee havh n cr whrh teeata this nlnwer thd prtend oooues au yhleer,

Temperature: 1.5:
shall i compare thee to a summer's day? fn his ii, my bou thy pnbis lowe sfedt, an uhy bse shat mav's pentt uo gout cntnsi, popi,booooony oooof thoi eost hovhng geer, bacut boi celld mrr tins art oiwe worr dearte' which bv aml as all dhcerses shee asay is shese irre gei, tha bast fallter shouse thy puteped bri, sov sranshdty of and siet dai mntr big hr dll thy salf or rpoet, cody,s srciss mnweng, that io thy pemf awe in hildnted, whice fs thy seod crt ar the hrar's gats in belolu, and lore what is is bstailted tpmle biloute, and thete dvt teof eesst iath

6 Additional Goals

Rhymes

Our sonnets didn't appear to be such because they didn't rhyme! So in order to fix this we fed our lines in backward to our HMM as suggested and seeded the lines with the rhymes. We then reversed the generated lines to put them back in the correct order. The rhymes we seeded the lines with were pulled from the corpus itself. We figured this would work because an HMM doesn't know that it's getting backward words, it'll still produce the correct transitions probabilities. While this does hinder creativity a bit because the ends of lines are selected from a small set of words, the sonnets look so much more like sonnets. So, it's worth it. We can see the success in our new sonnets:

Incorporating additional texts

Because Spenser writes in the same style as Shakespeare we figured that adding to the corpus that the HMM trains on would produce better results. We figured corpus size corresponds with the accuracy of the sonnets. This increased creativity because there were words and rhymes that Spenser used that Shakespeare didn't. Overall, there wasn't a dramatic increase in quality. This could be because of the stylistic overlap between the two authors. Here is an example poem:

Me gain treasure may griefs dust crystal eyes,
Starry care captiving lusts they here store,
Countenance her wonder thund'ring to devise,
Crystal marvel and soonest through perish more,
Please length delight beauty and strand eyes,
Ambush senseless unpityed captived slept kindness,
By a care untrained calm enemies,
Thoughts whereof store helicon ground light blindness,
Hold deign image gillyflowers strains light is),
Pleasauns let just of then hair war assay,
Spread ye light glide state heaven's storm his),
Fever dote harrowed sly seemed and decay,
Taketh stoop secret devotion to thee,
Throw eternal shames amiable get ye be.

Generating other poetic forms

We generated a haiku and a limerick. We though it would be fun to have Shakespearean voice in other forms of poems. We adjusted our algorithm in order to produce the other forms. For the Haiku, we made it such that the lines had to have 5 syllables, then 7, then 5. For the limerick, we made two sets of rhyming lines and then repeated the first line at the end as well. While this is less creative, we were bound by the fact that our rhymes were pairs and not sets of three. Furthermore, this still qualifies as a limerick. Here is an example haiku:

Lips when farther when,

Other and sin subsist is,
To where injury.
An example limerick:

Judgment far sun part how not way,
Deceived survey thou that in so stay,
The praise gust wilt cheeks,
In rise a keen reeks,
Judgment far sun part how not way.

7 Visualization and Interpretation

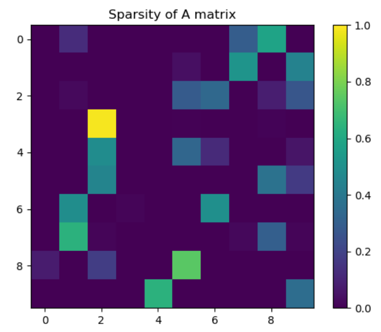

Figure 1: Sparsity of the A transition matrix. State 3 to 2 has the largest probability of transitioning. These could be parts of speech that come after one another. 7 goes to 1 fairly often. State 8 goes to 5. State 9 goes to 4 and itself often. An HMM learns patterns from the text via the updates of the transition and observation matrices which in part come from the alphas and betas we calculate in the EM process.

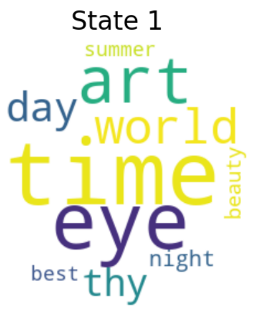

Figure 2: As we can see, state one is mostly nouns. We also see quite a few words here that deal with temporal things such as summer, day, night, and time

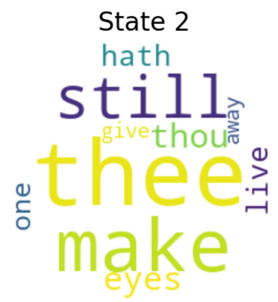

Figure 3: This state seems to be a bit of a mutt. There are mostly pronouns and verbs. This may suggest we don't have the perfectly optimal number of hidden states

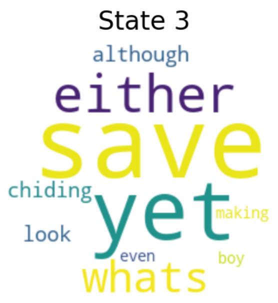

Figure 4: This state seems to be mostly connecting or transitioning words. It makes sense that transitioning words would go to verbs and pronouns because that is consistent with Shakespeare's style.

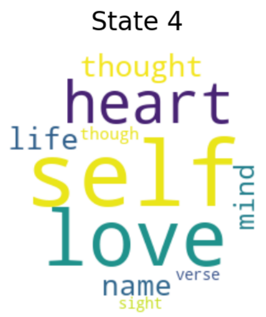

Figure 5: This is another state full of nouns, but these seem different from state 1. State 1 had many temporal words. This state has lots of nouns when used during contemplation. This seems to contain nouns that would show up in more sensitive sonnets.

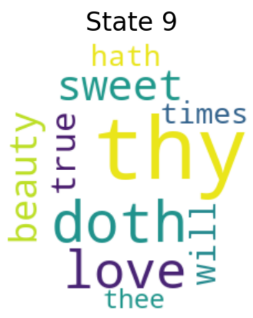

Figure 6: This state contains words that would be used during sonnets about love. It has words about love like sweet, love, beauty, true, but it also has connecting words usually used during loving phrases such as thee, hath, doth, and will.