Generative Modeling of Shakespeare's Sonnets using Hidden Markov Models and Recursive Neural Networks

Eli Sorey, Kun ho Kim

1. INTRODUCTION

Hidden Markov Models (HMMs) are effective learning frameworks for recovering unobservable latent state sequences that generate an observable sequence of data. HMMs are generative, in the sense that they can be used to sample data from the learned joint-distribution. These two properties make HMMs a great choice for the task of poetry generation. Shakespeare's sonnets have an underlying latent structure - rhyming, meter, syllable counts, and etc - which distinguishes his work from those of others. Hence, if we can train HMMs to learn such "hidden" constraints, then consequently it is possible to use the model to sample poems which mimic Shakespeare's work.
In this miniproject, we applied HMMs to the task of programmatically generating Shakespearean sonnets. These poems are structured as three quatrains of four lines each, followed by a two line couplet. The third quatrain is referred to as the volta. They have a rhyme scheme of 'ababcdcd efef gg', and adhere to iambic pentameter - ten syllables per line, with a meter pattern of alternating unstressed/stressed syllables. In this report, we provide a step-by-step explanation of how the HMM model was improved to evolve our poems into a version which finally sounds Shakespearean. Moreover, we interpret and analyze the learned transition and emission properties of the HMM in conjunction to our prior understanding of the latent structure behind Shakespeare's sonnets.
Long short-term memory (LSTM) unit based recursive neural networks (RNN) are also put to the test for generating Shakespearean sonnets. We attempt to add more variability to the sampling process by introducing a temperature variable to the soft-max activation at the output layer of the RNN. Sample poems generated by the RNN at different temperatures are presented. Furthermore, we qualitatively explain the differences in the poems generated by RNNs with those generated by HMMs. This report is organized as follows: In the methods section we describe the preprocessing techniques, learning algorithms, and sampling methods we used to generate poem with an HMM/RNN. In the Results & Discussion section we elaborate on the process of improving upon the sampled poems and comment on how well they adhere to the constraints of the Shakespearean sonnet. Moreover we analyze the learned transition and emission matrices. Finally the poems generated by a RNN are presented an compared to those generated by an HMM.

2. METHODS

2.1. Preprocessing

During the creation of the training set, lines from sonnets were tokenized into individual words. This was accomplished through the methods made available by Python's Natural Language Toolkit (NLTK). We decided to tokenize at the level of individual words because the primary structure of Shakespearean sonnets relies on the number of syllables in a line. Therefore, we wanted to maintain a sufficient degree of granularity to control the number of syllables in a line. We found it easiest to do this by considering one word at a time. In the process of creating our training set, we also set up dictionaries that mapped words in our training corpus to integers and vice versa. We accomplished this by creating a set of unique words in our training corpus and then simply numbering them. This simplified the training process by allowing us to work with integer "indices" for words instead of the words themselves. During this preprocessing step, we also opted to remove punctuation marks except for the comma. This was because an HMM was not able to learn how to appropriately place different punctuation marks. However, we found when only including the commas the HMM was able to appropriately place them. For the other markers, we intelligently reintroduce them during the post-processing stage.

Each training sequence consisted of words in a single line of a sonnet. This decision is related to the decision to tokenize each line into individual words - most of the structure in Shakespearean sonnets lies at the level of lines. It was most important for our lines to contain the correct number of syllables, and rhyme schemes occur at the level of lines (specifically the last word on each line). Therefore, we set up our HMM models to generate individual lines of the poem, rather than quatrains, couplets, or an entire poem. Once our models were properly producing 14 line sonnets with 10 syllables per line, we began looking for techniques to improve our results. Some research in Shakespearean sonnets revealed that the two quatrains (eight lines) tend to follow a similar style, while the third quatrain - the "volta" - and the last two lines - a couplet - have their own style. From here on words we will refer to the first two quatrains simply as "quatrains" and refer to the third quatrain as "volta" for clarity. We wanted this structure to be reflected in our model's output, so we decided to train a separate HMM on the set of first two quatrains, the set of voltas, and the set of couplets. This involved partitioning the training corpus into these parts and then training separate HMMs. The resulting model is therefore really a set of three models in which the first model generates the first two quatrains, the second generates the volta, and the third generates the closing couplet. In the process of separating the given sonnets into these sections, we noticed that sonnet 99 actually had 15 lines, rather than the expected 14. We dealt with this simply by removing it from the training corpus.

Another feature we wished to build into our model was rhyming. Our initial plan to implement this was to use the CMU Pronouncing Dictionary (CMUDict) as a ground truth for rhyme and force our models to generate rhyming words in the appropriate locations, while still generating the lines in a forward fashion. This approach makes sense in theory, but ran into several issues when we attempted to implement it. For one thing, CMUDict is by no means complete; it contains rhyme entries for on the order of 120,000 words. Our use case requires that any rhyming word must also be in the spirit of Shakespearean English; we attempted to enforce this constraint by requiring that any rhyming word from CMUDict must also be in the training corpus. Unfortunately, this led to a situation in which many of the words we encountered during generation simply had no rhyming word. At this point, we opted to try out the recommended approach of backward line generation. The first step in implementing this was to create a dictionary of rhyming words based on the corpus. This was straightforward, due to the strict rhyme scheme employed by Shakespearean sonnets. We simply parsed each poem into its ending words, and created a dictionary entry for every pair of rhyming words. Once this was done, generating rhyming sonnets boiled down to seeding the appropriate lines with words from the rhyming dictionary, tracking which words were used as seeds, and then using the rhyming dictionary entry for the seed word on the appropriate lines. For example, say that the rhyming dictionary entry for "mean" was "clean". We might seed the first line of a sonnet with the word "mean", and then when we get ready to generate the third line, we would use "clean" as the ending word. With this approach we were able to generate sonnets that correctly followed the rhyme scheme of Shakespearean sonnets.

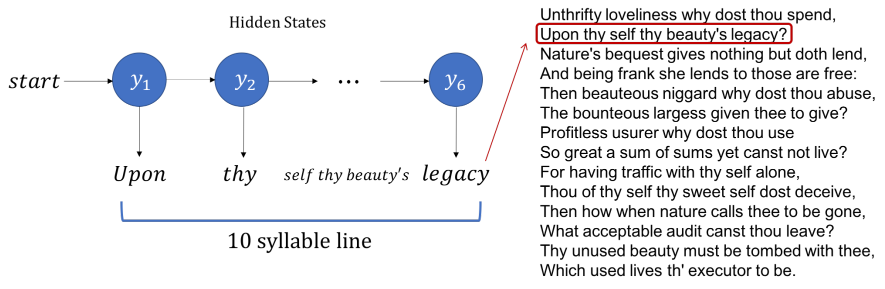
Hidden States
start
y_1
y_2
y_6
Upon
thy
self thy beauty's
legacy
10 syllable line
Unthrifty loveliness why dost thou spend,
Upon thy self thy beauty's legacy?
Nature's bequest gives nothing but doth lend,
And being frank she lends to those are free:
Then beauteous niggard why dost thou abuse,
The bounteous largess given thee to give?
Profitless usurer why dost thou use
So great a sum of sums yet canst not live?
For having traffic with thy self alone,
Thou of thy self thy sweet self dost deceive,
Then how when nature calls thee to be gone,
What acceptable audit canst thou leave?
Thy unused beauty must be tombed with thee,
Which used lives th' executor to be.

Figure 1: Overall pipeline of applying HMMs for poem generation

2.2. Unsupervised Learning Algorithm

Hidden Markov Models (HMM), falling within the class of generative models, seek to discover a set of latent states that emit observable features. In this project we train HMMs in an unsupervised framework to model the latent states y_i which emit the observations x_i corresponding to the sequence of words in a sonnet by Shakespeare. Fig. 1. shows the overall pipeline of applying HMMs for poem generation. We used the hmmlearn package for python which allowed us to train a multinomial HMM that takes as training input a column vector of sequences. As mentioned above, one training sequence consists of words in a single line of a sonnet converted into integers. The EM algorithm (Baum-Welch) was used to train the HMM in an unsupervised fashion. Two parameters were optimized during the training process: (1) Number of hidden states (2) Number of training iterations. The number of hidden states was determined through trial and error with visual validation. We assessed the quality of the output poem in terms of grammatical structure and coherency while varying the number of hidden states in increments of 5 from 10 states to 100 states. We determined that there was no visible improvement in poem quality when increasing the number of states beyond **30**. Moreover, the training time increased super-linearly with the number of hidden-states so there was clearly an incentive to determine the minimum number of states that provide comparable model performance to any higher-complexity models. The number of training iterations was determined by setting a threshold on the improvement in log-likelihood of the training data. When the likelihood improves by less than a factor of 10^{-5} of the improvement in the first epoch, training was halted. The number of iterations until convergence increased with model-complexity. (i.e. the number of hidden states used) For the optimum of 30 states that we determined, training halted around 50 epochs. As an additional goal, we attempted to train the hmm in a semi-supervised fashion by partially labelling the hidden states according to grammatical structure. For example a state that represents an adverb should transition to a state representing a verb and so fourth. However, this process was extremely labor intensive and we were not able to see visible improvements in the generated poems. For future reference, we postulate that crowd sourcing this task on Amazon MTurk to gain a sufficient amount of labels would increase the quality of the model learned in a semi-supervised fashion.

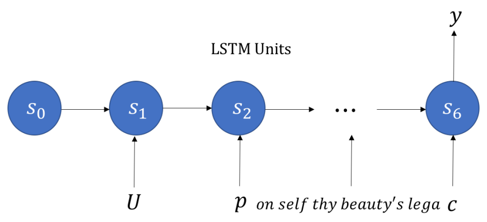

Figure 2: LSTM networks for poem generation

2.3. Generative Sampling

As an HMM is a generative model, we were able to sample fresh poems using the learned state transition and emission properties. Our sampling process takes place in 3 steps: (1) Choosing a word from the rhyming dictionary as the initial seed (2) Reverse sampling a line from the HMM while adhering to the 10 syllable constraint (3) Post-processing the poem to add punctuation markers and more. The first word (which corresponds to the last word in a line) was sampled from a rhyming dictionary at random. An initial hidden state was randomly sampled from the start probability distribution A_{start}. From here on we repeat the process of randomly sampling a word and transitioning to a new hidden state in accordance to the transition properties of the current state. The number of syllables were tracked through a syllable counter function from the TextSTAT package for python. If the number of syllables surpasses 10, then we discard the last sample word and re-sample from the same hidden state until we get an exact syllable count of 10. After the poem was generated, we post-processed the text by adding punctuation and capitalization. The beginning word of each line was capitalized along with the pronoun "I". For the punctuation, we recall that all but the commas were removed from the training set due to the fact that the HMM was not able to appropriately model the location of punctuation marks in a sentence. Hence, we added other punctuation marks by sampling them from a probability distribution which was generated by simply counting the occurrences of each punctuation mark at the end of a line in the training data. (Most punctuation markers only appear at the very end of the line) Typically we saw that commas were most frequent and thus our poem generally embodied this characteristic. Moreover, we noticed that the very last line of the poem always ended with a period. Hence, this property was added to our poems in the post-processing step. We present the poems sampled using this method in section 3.1

2.4. RNN Training

As an additional goal, we attempted to train LSTM networks to generate poetry. (Fig. 2) Recursive Neural Networks (RNN) with LSTM units utilize memory gates in order to further propagate gradient signals while avoiding the vanishing gradient problem prevalent in most deep networks. Instead of using words as tokens, we decided to use individual characters. The training dataset was partitioned into strings composed of 60 characters. The reason for choosing 60 was because all lines in Shakespeare's sonnets did not exceed 60 characters. Hence, we sought to allow the network to learn approximately when to end a line by outputting a new-line character. A slide of size 3 was used to move across the training data and generate multiple input sequences that the network can be trained on. We utilized the Keras deep learning package for training. At first we attempted to train using a small number of hidden units (128, single layer) on just the Shakespeare's sonnet dataset alone. We immediately realized that the model was not able to learn a representation sufficient to generate interpretable poems. However, using more hidden units on this small dataset also did not yield understandable poems. Hence, we sought to collect more training data of poems written by various other poets. This decision came at the cost of giving up on training a model to learn the format of Shakespeare's sonnets, but it allowed us to generate decent poems with different form. We collected the works of three other poets: Sylvia Plath, Edgar Allen Poe, and Pablo Neruda. The works of these poets were readily available online and we simply converted them into a text file that can be processed by python. The total training set then had around $500,000$ characters which was around 5 times larger than the size of the Shakespeare dataset alone. By visual validation of the poems, we determined that a 2-layer RNN with 512 LSTM units per layer each was sufficient to generate decent poems after training for 50 epochs. Training took around 16 hours. In order to introduce more creativity and variability in the output text sampled from a trained LSTM network, we incorporated *temperature* into the prediction of the model which takes a modified soft-max activation of the predicted emission probabilities in the following way:

$$P(x_t = c^*|x_{1:t-1}) = \frac{e^{p_{t,c^*}}}{\sum_c e^{p_{t,c}}} \rightarrow P(x_t = c^*|x_{1:t-1};\tau) = \frac{e^{p_{t,c^*}/\tau}}{\sum_c e^{p_{t,c}/\tau}}$$

where τ is the temperature, and p_{t,c^*} is the probability of outputting the character $x_t = c^*$ at time t given the previous emission sequence $x_{1:t}$. We present sample poems generated with this model in section 3.3

3. RESULTS & DISCUSSION

3.1. Evolution of Poems

An example of a poem generated by our first naive working model is shown below. This text was generated from an HMM simply trained on the dataset line by line without any preprocessing besides converting words into integers. No post processing besides the syllable count was performed.

> by to this expense flame less pen upon errors adjunct
> still . and reason it , gives my purpose before
> , in keeps my fist did time . shall the
> well-contented roof to thee too beauty may staineth , more
> fiend , then keeps the prime . though thou abuse
> accumulate , these these to art things blot . the
> lines bark our gay . in praise cover part you
> lend , there be is not to tombs in sweet
> , and perfect'st informer , by single shalt fortune 's
> attainted days which i find . if this i usest
> ? or i common fits . for winters be ,
> which who acquaintance eye in the gentle lies changing heart
> ? from thee shame , at to art attainted ,
> that thou do , or i better her other

As you can see, this model is not intelligent about placing punctuation, and does not get the syllable counts exactly right. The first issue was caused simply by the fact that we did not remove punctuation before training the model. The reason for the latter issue is that our method of counting syllables at this stage was imperfect; the method call returned a float and we then had to round that value and infer how many syllables were actually in the word. At this point we had not ironed out the syllable counting logic perfectly, so some of the counts are off.

Here is an example of a poem generated by an improved model:

> If I thy short was walks and height with a lords
> That my scythe, want thou alike not your
> We not in land loves in muse and that
> And thou but her as and errors nor be side
> Now do weakens to give only is and
> But husband tongue grow forbidden onward
> O think, creatures thee wherein youth knowing
> After the earth to tell idolatry false
> Whilst yet thy mouths of forged bloody grow live
> Although live are thy truth of kingly
> Have in those looked in physic to
> Let his them hate view most ending do I
> He on these odour, hath authority
> From me, are art that my living disperse

At this phase, we had begun removing punctuation prior to model training, and had opted to split the training corpus into leading quatrains, voltas, and couplets, as described under the Methods section above. However, there was still a bug in our syllable counting logic that meant our syllable counts were sometimes off by one. There is also no attempt at rhyming, and we had not yet re-added punctuation. (Besides the commas which are included in the training set)
Finally, here is the poem that we used for submission:

Youth face to come more prizing in spirit,
Seen bosom would she from what quality,
Rebel to pleased doth days most sun look hit.
Will heart shall thou when and astronomy,
Will book such drown glass the burn or turned,
Dear thou each, misuse that lour'st should but urge,
Muse those to not nature keep or burned,
Eye of map gentle are princes if purge.
Only true likeness that looking and know:
Always I might papers his know o store;
My determined before eyes thou now show,
Done cherish gave them me civil that more,
My forsworn straight and effect this bear got
Know of single I, perceiv'st you but not.

At this point, we had switched to a more straightforward library method for counting syllables, overhauled our syllable counting logic, implemented intelligent punctuation placement, and integrated rhyming as described in the Methods section. The result is that our poems from this model consistently had ten syllables per line and properly followed the rhyme scheme of Shakespearean sonnet.

3.2. Learned Model Interpretation and Visualization

3.2.1. State emission properties

We looked at the top ten words emitted by the states of each of our three models - leading quatrains, volta, and final couplet - in an attempt to better understand the semantics of each hidden state (see Tables 1 through 3 below). As the precise state numbers do not contain any particular meaning, we simply labeled the states in ascending order. Overall, we may observe that the likely words for each trained model represent different tones as expected. For example, the quatrains model contains monotone words such as fine and pilgrimage while the volta model has more descriptive adjetives such as duteous and deformed'st. The couplet model contains more mystical words such as creatures, kingdom, and love-god.
By merely looking at the top 10 words it was confidently assign human-interpretable meaning to hidden states in an HMM. In particular, the top words for the leading quatrain model (Table 1) seems quite arbitrary. However, we can still see some groupings such as in state 1 which seems to primarily contain adjectives. In the following transition property section we show that by looking at more words it is possible to discover a number of states that contain a majority of words that are a certain part of speech.
We see that many high probability words in the volta model are adjectives. As adjectives are used primarily in descriptive language, we hypothesize that Shakespeare may use the volta to delve into descriptive language about the topic of his sonnet. The word volta means "turn" in Italian, and this section of the poem represents a turn in the thought or argument of the piece. Perhaps Shakespeare uses further description in the volta to shift the audience's attention to a different facet of the object at hand. It may also be the case that in general poems contain more adjectives than other parts of speech.
Finally, the words most likely to be emitted by the states in the couplet model appear to have relatively more one syllable words. Since the couplets in Shakespeare's sonnets follow a different rhyme scheme and are used to conclude the pieces, it makes sense that they may also follow a modified syllabic structure. This is not to say that these lines deviate from the ten syllable structure, but rather that perhaps syllables split along words more exactly in the couplet section than in the volta or leading two quatrains sections. We also see that state 12 seems to contain more nouns than other states. We believe the HMMs are able to learn such properties of texts since many transitions between words in the training sequence adhere to grammatical constraints. Hence, during the M-step where a summation of marginal probabilities take place, the grammatical constraints will drive the probabilities up for more frequently observed transitions. However, since the data is labeled, it is also not perfect as we can see from the tables.

Most Likely Words for Leading Two Quatrain Model				
State 0	State 1	State 2	State 3	State 4
grounded	reserve	censures	vassal	vassal
actor	robbed	vacant	reserve	grounded
quicker	crooked	weigh	robbed	actor
pilgrimage	resembling	lest	crooked	conquest
fine	looking	vanishing	resembling	razed
broad	thus	reserve	looking	pilgrimage
essays	admit	fortify	thus	bristly
sway'st	creep	sealed	admit	forgetful
reserve	o'ercharged	robbed	creep	fine
poesy	crawls	else	o'ercharged	broad

Table 1: Top ten most likely words for the leading two quatrain model. Other states omitted for brevity.

Most Likely Words for Volta Model				
State 5	State 6	State 7	State 8	State 9
delights	delights	delights	mistaking	delights
gaze	tenth	familiar	stern	gaze
nightly	building	scope	deformed'st	nightly
familiar	duteous	admired	holds	put
put	bud	instant	building	lest
lest	slandering	slandering	duteous	shallowest
instant	ruin	ruin	bud	monument
carve	orphans	root	else	building
bud	resembling	resembling	lily	duteous
slandering	lily	three	meditation	carve

Table 2: Top ten most likely words for the volta model. Other states omitted for brevity.

Most Likely Words for Couplet Model				
State 10	State 11	State 12	State 13	State 14
odour	feil	nightly	odour	odour
feil	powers	powers	flattery	matter
powers	pale	creatures	feil	nightly
dream	ten	whit	offices	feil
pale	creatures	faster	powers	powers
most	name	grounded	dare	dream
ten	huswife	love-god	pale	dare
creatures	rebel	rebel	creatures	pale
faster	other	conquest	rebel	cheer
love-god	conquest	king	dumb	kingdoms

Table 3: Top ten most likely words for the couplet model. Other states omitted for brevity.

3.2.2. State transition properties

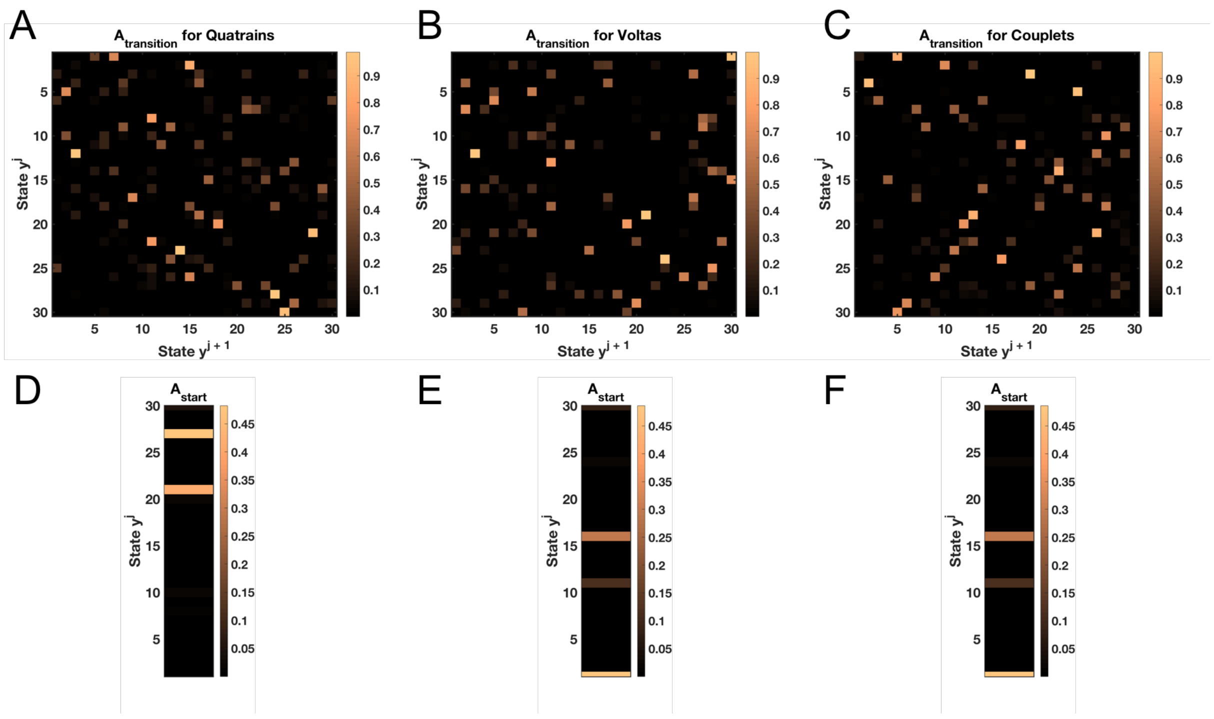

Figure 3: Overall pipeline of applying HMMs for poem generation

Fig. 3. Shows a heat map of the learned transition matrices. The color intensity ranges from 0 to 1 and represents the transition probabilities. Note that the transition matrix was transposed to represent the forward transition probabilities since each line was learned in a backwards fashion. Moreover, when we examined the top 10 likely emissions for each state, there wasn't a clear majority as to what class of words a particular hidden state contains. However, by examining 100 words for certain hidden states, we were able to see some correlation to parts of speech. For example, there were states found with a majority of likely emissions being verbs. With that in mind, let us examine some of the key properties of the state transitions that were learned during the unsupervised training process.

- For quatrains, voltas, and couplets there always exists states that exclusively transition to one other state. For example, Fig. 3A. shows that state 21 will almost always transition to state 28. From examining the mostly likely words that these states will emit, we saw that this transition models the adverbs $\rightarrow$ verb transition. (Note that the numeric labels for the states are stochastic in the sense that they change during every training session. Hence these labels may be different from the ones presented in the previous section) The same transition was observed for the voltas in Fig. 3B. where state 13 transitions exclusively to state 11 and for the couplets in Fig. 3C. where state 3 transitions exclusively to state 20.
- Similarly to the adverb $\rightarrow$ verb transition, we found high probability state transitions between pronoun $\rightarrow$ verb, noun $\rightarrow$ verb, and verb $\rightarrow$ noun. The exact state labels are omitted for brevity.
- On the contrary, we also discovered that grammatically unlikely transitions such as verb $\rightarrow$ verb had low probability transitions. We can observe this from the fact that the diagonal of the transition matrix is generally dark.

- There were multiple states corresponding to the same parts of speech. For example, verbs were present in state $21, 15$, and etc for the quatrains.
- The start probabilities show that sentences will most likely end (recall that lines were generated in reverse order) with verbs or nouns.
- Generally the quatrains transition matrix finds clearer distinctions in the semantic meaning of each hidden state. This become conspicuous when comparing the start probability matrices in Fig. 4D, 4E, and 4F. We see that the quatrains start probabilities are most "definite" in the sense that they have discovered a clear pattern of what parts of speech tend to end sentences. Moreover they match well will the fact that Shakespeare will most often end his lines with nouns or verbs. On the other hand, the start probability distribution for voltas and quatrains are more widely distributed. We attribute this to the fact that the HMM for the quatrains was trained on twice as much data as the voltas and four times more data than the couplets.

3.3. LSTM poem generation

The following are three poems generated by the LSTM network at different temperatures. ($\tau = 0.2, 0.5, 1.5$)

Farewell

How loveling at the truth
And the wanderes hate and polumn's slow,
Sick beauty's thoughts of thee what were worsh in the sky,
And with my missticate possiquingled
What appracing the winds
And yet merrousing their pipting and the realien
The stars were the sun is it
Were on the mighty goodly gives to the well you whits away,
For long them to thee.

Angels

The is it not with the univers' shive a lide
Fight more by poidows visit he hours trust:
O and all her poirs of angels heavenly blindon
Marizant for me for thie.
Of the most seccouss of all stay,
Of the angels grave upon the staterord there appears to the deck of ant fill!
For that a feeling savizy of surfect without soin;
Is not that the dark angel–sight shall chase autwered in the 'southern loves in happy
He came ther pare that I in the with conqueror worm.

Flight of Heart

Parring the flight of heart to this more in
Facumuse that hangs upon, the earl hath hold the abserve of my heart,
And where the well I sap rid cake, is med amid mistameth,
Sorrible, flaction but the sea
Will enwers thee kill.

Yet is in–the contaction?–beauty of the night-wainy of thranting, drymbering
There is a holion shant
Scents nones–the comite and ther blue joors still in this beauty,
All this aring with the spothes, to be subeburate length,

The first poem was generated at $\tau = 0.2$, the second at $\tau = 0.5$, and the third at $\tau = 1.5$. The output of the LSTM network was trimmed off at 9 lines in order to show poems with consistent length. Since the training set was composed of poems from various writers, the RNN was unable to learn a fixed length for each poem. We can see from all three poems that in general, the RNN is capable of learning proper grammar. Moreover, it's impressive to see that for low-temperatures, most words are spelled flawlessly despite the fact that the network was trained on character tokens. Compared to the HMM model application, we found that RNNs were able to model the grammatical structure more accurately and place punctuation markers appropriately. As some poems in the training set did not have rhymes, the RNN did not model the rhyme scheme. (We however saw instances of rhymes being learned when only training on the Shakespeare data set) One major drawback was the amount of training data necessary and the amount of computation time needed to generate a decent model. As the RNN was fed character based sequences, it required much more training data and time.
The variation in temperature yielded interesting changes in the output poems. We can see that as the temperature increases, the accuracy of the output decreases in the sense that words are mis-spelled more frequently and erroneous grammatical structures are more prevalent. However we can also observe that the style of the poem is more hybrid and unconventional then the previous poems. For example, the third poem "Flight of Heart" combines the diction of Shakespeare along with the tone of Edgar Allen Poe. With more training data and time, we expect to be able to exploit temperature to generate creative poems that are also grammatically accurate.

4. Conclusion

From this mini-project, we gained a better sense of the benefits and drawbacks of generative models and unsupervised learning. Our team was able to generate interpretable sonnets that mimic the structure and tone of Shakespeare by training a HMM in an unsupervised setting. We met the standard goals of naively generating a 14 line sonnet that was understandable by a human by using 30 hidden states. We achieved/attempted various additional goals including: 10 syllable formatting, rhyme scheme, post-processing, and semi-supervised learning. The learned transitions and emission properties were analyzed in depth. Moreover, an LSTM network was successfully implemented and compared to the HMM in terms of the quality of the poems generated. We show the feasibility of generating more creative poems by incorporating temperature based soft-max activation on the output layer of the RNN.
As future goals, we believe implementing the iambic pentameter scheme will be an interesting task. We would also like to implement HMMs in a supervised learning setting by utilizing crowd-sourcing services such as Amazon MTurk to obtain the labels. Furthermore, we are curious to see the performance of an RNN trained on a larger dataset for a longer time with GPUs.
Both authors of this report contributed equally to both experiments and write-up. Eli primarily worked on implementing the additional goals for improving poem quality. Kun ho developed the code for the naive poem generation and ran the RNN experiments.
Lastly, we would like to thank the TAs and the instructor for organizing a fruitful learning experience. We hope this tradition will carry on in future years of CS 155.