Machine Learning & Data Mining
Miniproject 3

Caltech CS/CNS/EE 155
Released March 6^{th}, 2020

---

1 Introduction

- **Group Members**
  Emily Park, Megan Tjandrasuwita, Elizabeth Yam
- **Team Name**
  Yoshi's Island
- **Division of Labor**
  Emily and Megan implemented pre-processing methods to tokenize the words and create the syllable dictionaries. Elizabeth implemented the unsupervised Baum-Welsh technique. Emily implemented the Hidden Markov Model sonnet generator and Megan implemented the Recurrent Neural Networks sonnet generator. Emily implemented incorporating Spenser's sonnets, rhyming sonnet and haiku generators. Elizabeth and Megan implemented visualization.
- **Github Repo**
  https://github.com/eppark/Shakespearean-Sonnet-Generation

2 Pre-Processing

We used individual words as tokens and split up the data by lines, while making sure not to count line numbers. We made sure to not include punctuation, including commas, semicolons, parentheses, etc. since these can interfere with our syllable counts (particularly with Spenser's work, since we were not given an official syllable dictionary for him). We also processed the contents of the syllable dictionary by storing separate dictionaries for the typical syllable counts of words and the different syllable counts for when a word appears at the end of the line. Later on, we also did the same preprocessing for the Spenserian texts. In the case that we were unable to use NLTK's CMU dictionary for some of the words in Spenser's text, we avoided including the entire line in our training set since we trained on a line-by-line basis.

3 Unsupervised Learning

Instead of using a package, we used the unsupervised HMM implemented in set 6. We chose 10 hidden states because there are 10 syllables in every line of a sonnet, so there can be at most 10 single syllable words. Because an input sequence is a line of a sonnet, we have 10 states corresponding to the position of a word within a line.

4 Poetry Generation, Part 1: Hidden Markov Models

Since we trained our unsupervised HMM line-by-line, we generated our sonnet line-by-line as well using 100 iterations. Since we don't use our syllable count data in our training, we needed to manually ensure that the lines we generated were 10 syllables each. As a result, the quality of generating poems in this naive manner is very poor, since there is no sense of rhyme, rhythm, or syllable count based on what a sonnet should be as we do not incorporate this data in any way. They do not make much sense, since the lines are generated without meaning, but they do retain Shakespeare's original voice as we used his sonnets

for our generating process, so the overall vocabulary and sentence structure seems fitting. Training with more hidden states made the quality a bit better, but past 10 states the quality seemed to taper off. Since the HMM used probability of words occurring next to each other to generate the sonnet, the lines tend to follow a normal grammar structure (ie. not many verbs next to verbs or adjectives next to verbs). Punctuation and capitalization were not used when training, so our HMM did not incorporate these characteristics. We incorporated capitalization after, in our generating process.

Poem generated with 5 hidden states:

Proof thoughts to what art we liberty to
Thou joy have that mine hand of have this green
Home meditation let but thy lov'st to
To so beard love of to my it tenants
Treasure our him warm be number deep
Knowing dust thou tyranny pitch beauty
Decay love should like your monuments wits
May those was shows influence i treasure
Or when than on slay hang as o me thou
The but false when as her wounded abuse
A pride i art a from free age soul thou
Much book set his will star like and world and
Rare he bosom saw and thou maintain with
Friend let me each eyes is my same of fist

Poem generated with 10 hidden states:

To with beauty that more i store so heir
When wilfulness goddess and love and threw
Me all alone living shore that present
Argument thy thou which once never there
More best bond of less thy muse with the have
Ill i rearward spend am no breath and taught
Before hide brav'ry art not bright nothing
Trenches but death when whether how see you
Or thee form seem thou were noon o preserve
As altered which breath die image for catch
Should wish i o'erlook i afterwards have
So come of made right blushing both a thou
Two enemies and prove form doth and time
Mind fast but thy heart and shape what self die

5 Poetry Generation, Part 2: Recurrent Neural Networks

We implemented a character-based LSTM using Keras. We tuned the number of nodes in the LSTM layer, the batch size, and the number of epochs by minimizing loss. For the size of the layer, we consid-

ered values of 135, 145, 165, 185, and 195 and found that 195 gave the lowest loss when holding all other parameters constant. For the batch size, we considered values of 32, 40, 75, 100, and 128 and found that 32 gave the best loss as well. To ensure that we used enough epochs to train the network, we passed in a EarlyStopping callback when fitting our model, which restores the best weights from during training as well as stops training when no improvement in loss is seen over a period of 8 epochs. Using the given seed, the sonnet produced using a temperature of 0.25 is:

> To their that woold moth proud of not dost
> Creight io not do fle the part, te dir resees,
> Whe evrence self not forr shall see prife,
> The icjeed that dese faire, by this theas spuef the lars,
> An y this will sovely self at of me
> Art it the fair will soader's all herr grain,
> That whold moth it one efund eyer to them thee,
> Whered I am the spick the lappertiens.
> For al sweet heart what ot thou thy strongh the cold,
> But the ulow brt delle, and I heart so praise,
> Ther your rese conce as neathl wate heart.
> Whinh I chist the gase uf spprip to and the wild ene should expies,
> Which hady mose on thoughts conqurests didvite
> To what where farr yet the farthes sandst creas the sight, that thou meast I live ene hould self ate,

The sonnet produced using a temperature of 0.75 is:

> The ridy the ham the can to prree?
> For wruth the ter lave that chis not your least,
> But muting no strongs of this sigrt and my uppast,
> Or east the place shall gave the leasure,
> Now canter grace as time loke thee light,
> But that you fill seep in thee themeld,
> Why alf love thou and the Makes becume thy will, see:
> inllosing now stall tommed thy sweet ste,
> And as the priend bee than the beld:st dweals,
> Whin illon thre thou fiemands shall fove me filf,
> I for their reserd devering thee very the,
> And the end acd dath ughord to gheat,
> That why show still when that I thill so,
> So should mast aghar my some with me,

The sonnet produced using a temperature of 1.5 is:

> Athimine ort to the 'scepa self,
> And sight to gell and surnd agh long and wint

The patile of my deld,
And shall to chastakence as is men's holy lears,
That in his conqueld by sweet dead seched,
The live aw it time buly lot shame still,
Thy foor a formertert thre fave heaven's pord,
And then that sulf such suching that to crosimione.
Then cand then conerey, and athin thou from,
Astivit thy part,
an y grost consting no powe,
Sing an all they love is racker phase,
The pistance should be onceaso, I dost infeced,
Wo' rain shilot Thou be givess modr.

Overall, while the vocabulary of the RNN is unconventional due to using a character-based model, there is some semblance of basic sentence structure in the sonnets, as in phrases within lines make some grammatical sense. In all three cases, the RNN also learned to capitalize the words at the beginning of the line, as well as outputs punctuation frequently at the end of lines. The sonnet produced using $T = 0.75$ is the closest to adhering to the structure of a sonnet, as its line breaks occur roughly after 10 syllables are outputted. Because many of the words are nonsensical, it is difficult to determine if there is some semblence of meter, and none of the sonnets seem to have a rhyme scheme.

The quality of the poems produced by the HMM is superior to that of the RNN sonnets, as it is difficult to decipher meaning from the RNN sonnets due to the RNN's poor vocabulary. The runtime of the RNN is longer than that of the HMM, as each epoch took on average roughly 20-22 seconds. The model with $T = 0.25$ took 86 epochs to train, with $T = 0.75$ required 81 epochs, and with $T = 1.5$ took 138 epochs. On the other hand, training a HMM took only a few minutes with 100 iterations.

The sonnet produced using $T = 0.25$ seems the most confident in its output, resulting in some irregularities such as the abnormally long last line. Furthermore, much of the vocabulary is consistently nonsensical throughout the first poem, as opposed to the second sonnet where some lines make more sense than others. These observations make sense as inflating the values that are passed into the softmax function makes the RNN more likely to activate the output layer, so the model is more confident in its decisions, though most decisions seem to be faulty. The sonnet produced using $T = 0.75$ seems to be the most structurally and semantically sound of the three sonnets, but even so, there is little meaning to extract when looking beyond short phrases of the poem. Finally, the sonnet produced using $T = 1.5$ seems to have made the most mistakes out of the three, as the length of its lines are overall shorter than expected, and not much else makes any sense besides the first words of the lines. The outputted words seem to stray further from proper sentence structure compared to the other two sonnets since the RNN sacrifices correctness for diversity of output.

6 Additional Goals

Our unsupervised HMM does not naturally create sonnets with 10-syllable lines each. Thus, in our line generating function, we ensured that we would only produce lines with 14 syllables each; if there were

more or less, we would regenerate a line until a success. Thus, in this way we were able to generate sonnets with the correct syllable counts.

Since all 139 of Spenser's sonnets in the Amoretti follow the same rhyme scheme and meter as Shakespeare's sonnets, we were able to incorporate Spenser's sonnets in our training data as well. We use similar pre-processing methods to tokenize each line into words, remove punctuation, and add lines into our training data. To get the syllable count of words, we used NLTK's CMU dictionary for every word. If a word did not exist in the CMU dictionary, we omitted the line entirely, since our training relies on entire lines. Then, we trained on the data using our unsupervised HMM algorithm with 100 iterations and 10 hidden states. With this added training data, the generated poem sounds a lot better and has better sentence fluidity/grammar.

Poem generated with both Shakespearean and Spenserian text:

Seeking strains let spirits ladies smiling
Fixed so away receive quill dust that love
Thou locks filching spirits chace wilfully
Believed sharp'st pitch melts catch hues scythe ensconce
Couldst then from that she admire workmanship
Trifles seek fame joyous attire dust
Forms sacred wear they fly my days to as
Sharp'st onwards dangerous titles pleasure
Well-tuned heavens she cause be spirits
Cordials servant had and pay resort rough
Mercy and ye prison it heaven sought
Painful raging overcast hunts pupil
Inferior love true with and created
And that thing lark called resort sharp'st bay called

We also incorporated rhyme by creating a rhyming dictionary from Shakespeare's work, since all of his poems follow an abab...cc rhyme scheme. We omitted poems 99 and 126 from our rhyming dictionary, however, since poem 99 has 15 lines and poem 126 has an aabb rhyme scheme, which differ from his other works. From this, we randomly picked rhyming word pairs and generated our lines backwards to create rhyming sonnets with Shakespeare's usual abab...cc rhyme scheme.

Poem 1 generated with a rhyme scheme:

Be flattery ever takes delight smother
Fresh to that loves ill we leave majesty
Wretch then in thou o fair seem and other
Summer necessary care them whose eye
As benefit all thee princes away
I heaven art self that in task ground slow
Be die love blanks the rondure thee can stay
Thy thee to say is youth your his of woe
Will he they and content nature with brought

It a beloved and sick face my lays
Learn death return do once now should what thought
Art of everywhere greet in and me days
Carved can should stand stern quiet guilt pen fame
Of deceived can whate'er welcome name

Poem 2 generated with a rhyme scheme:

Dumb from manners by that how worse couldst verse
Part his where scorned it make each can is play
Hair true year bareness compare some rehearse
Great love is after face worthier away
Their live such thou that how doth thou friend sight
Heart my eyes plagues pent blood thou their high black
Then says scorn name my with oppressed of night
Power world the are name things one is hue lack
Just to receives love the when is desire
Thy wretch why poor repair see men to left
Day the left to turned on of tend conspire
Beauty i wilfully sparkling bereft
Lovely three sovereign my thou back a prime
Must is and level thou it i holds time

We can also easily generate other forms of poems. By changing our emission generator conditions, we incorporated generating haikus of three lines with 5,7,5 syllables each.

Haiku 1:

Near them respect to
Changing love to apparel
Now so then o'er-read

Haiku 2:

Of scope than alone
And dead so thou then else will
Disdaineth if tied

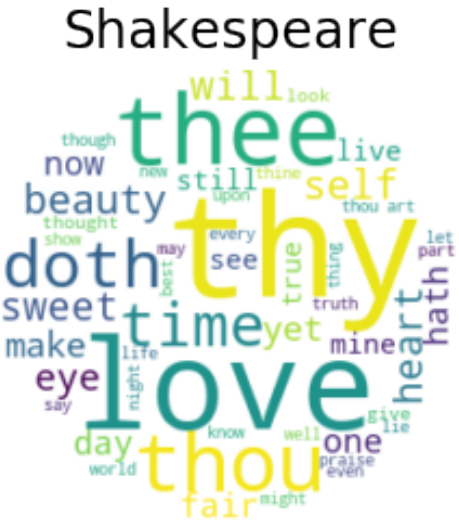
Haiku 3:

Garments it on thee
Favour weeds more better say
Belied majesty

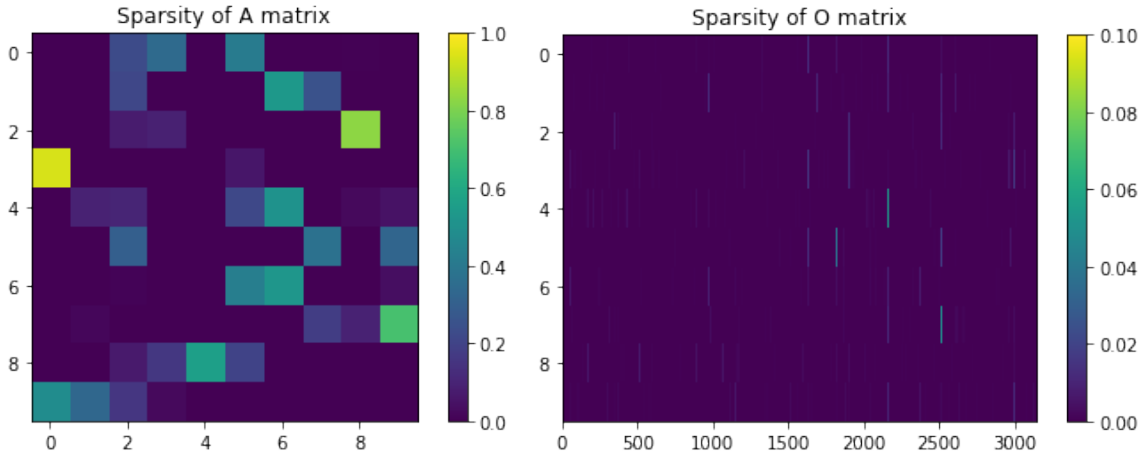
The haikus do not seem to mean much, which makes sense, since we are using Shakespeare's work to create them, and Shakespeare's sonnets are much longer than haikus. However, we are still able to produce haikus that have some correct sentence structure.

7 Visualization and Interpretation

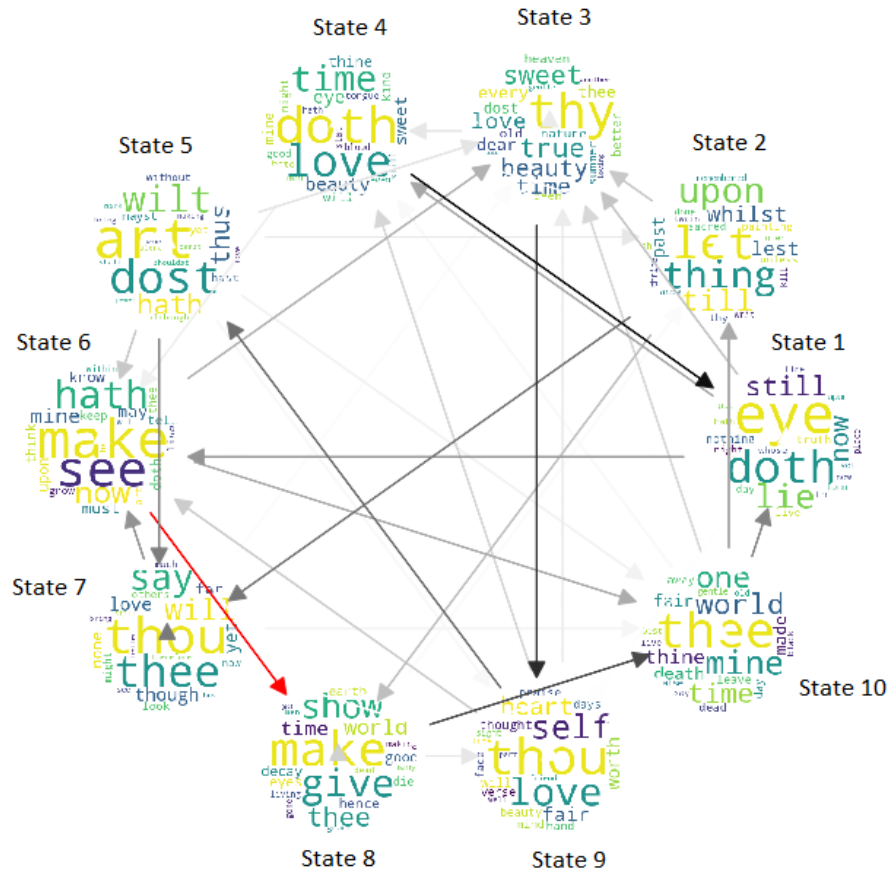
We can visualize Shakespeare's text as a word cloud where prominent words are displayed bigger than others. As evident in the visualization, Shakespeare's poem has short nouns (e.g., "thou" and "love"), determiner ("thou"), and verb (e.g., "love"). This suggests that the Hidden Markov Models may be training on such common groupings (i.e., short nouns, determiners, and verbs).



The visualized sparsities of the A and O matrices show that for every state there are a quite a few states that have equally high probability of coming next. As the subsequent word clouds of the states shows that each state contains short nouns (e.g., "thee", "thy", and "thou") and short verbs (e.g., "make", "see", and "let"). Due to the similarities of each state, its reasonable that there are quite a few states that have equal probability of coming next. However, for the observation matrix almost all the words are unequally likely to come from a state except for a few words. The sparsity of O can be explained by his infrequent use of specific words (e.g., "mutual ordering" from poem 8) but commonly used nouns such as "thou" and verbs such as "make". The sparsity in the visualization can also be due to that there are greater than 3,000 words that a given state can transition too, which lowers each observation's probability.

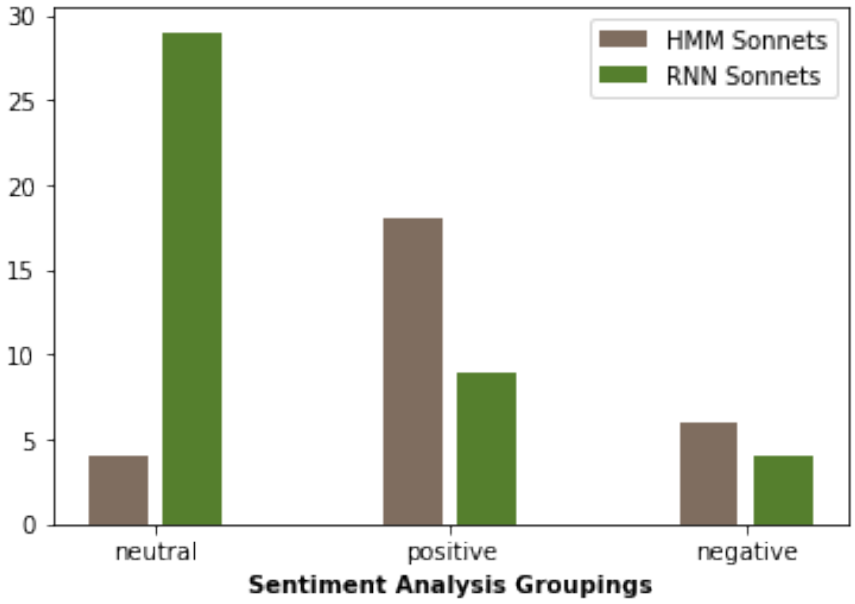


The connection between each state can be more clearly seen in the mapping below where darker arrows described a higher probability of transitioning from the state at the tail to the head of the arrow. As mentioned before, each state contains a noun and a verb. In addition, each state transitions follows corresponds to a noun and verb pairings that Shakespeare commonly uses, such as ”thy self” and ”art thou”. We also see the repeated appearance of words in different state, such as ”thou” in state 7 and 9, that reflects the prominent words used in Shakespeare’s poem and the common phrases ”thou art” and ”art thou”. Thus, we can see that HMM model takes into account both phrases ”thou art” and ”art thou” (which are the reverse of each other) and the repeated appearance of ”thou” by including it in more than one state.


And is immortal if thy thou is in
State 4
State 3
State 5
State 2
State 6
State 1
State 7
State 10
State 8
State 9

State	Top 10 Words	Common Features
State 1	let, thing, past, till, lest, upon, whilst, thy, thrice, therefore	Prepositions, transitions, and time/event-based words
State 2	doth, eye, still, now, truth, lie, nothing, live, life, upon	Some words are verbs (e.g., ”live” and ”doth”) and some nouns (e.g., ”live” and ”life”)
State 5	make, hath, see, now, may, mine, art, know, upon, tell	Verbs
State 8	thou, love, self, heart, fair, well, worth, praise, mind, will	Words related to love and the soul
State 9	thee, time, world, mine, one, thine, death, fair, leave, best	Quite a few words are possessive (e.g., ”mine” and ”thee”) and a few are with departing (e.g., ”death” and ”leave”)

We also ran the NLTK's VADER sentiment analysis tool, which is tuned on social media data but also works on other types of texts. We ran the tool on the same three sonnets listed in the ”Additional Goals” section. The first sonnet is generated using a HMM trained on Shakespearean and Spenserian texts while the latter two are generated with rhyme scheme. Then, we analyzed the three sonnets generated by the RNN in part 5 using the same temperature values of $T = 0.25, 0.75,$ and 1.5. Applying VADER to analyze each line of the poems generated by the HMM and RNN, we compare the results of the two different models below.



There are three possible classifications of poem lines: neutral, positive, and negative. For the HMM poems, the most prevalent binning is positive, whereas for RNN, the neutral classification occurs the most

often, greatly outnumbering the positive and negative binnings. This makes sense as VADER is trained on social media text, where proper vocabulary and sentence structure is used, whereas most of the RNN output fails to follow conventional English rules. With the RNN lines, VADER struggles to make a classification decision, as opposed to the HMM lines, where the rate of VADER settling on the neutral classification is much lower. These results imply that the sentiment analysis results are less interpretable for the RNN output, but it is possible to conclude that the HMM has a tendency to generate poems with a positive tone.