Machine Learning & Data Mining
Miniproject 2: MovieLens Visualization

Caltech CS/CNS/EE 155
Released February 16st, 2024

---

1 Introduction

- Team name: Butters
- Team member names: Etienne Casanova, Jack Myles, Ishaan Mantripragada
- Work Division: Etienne worked mostly on the HMM, Ishaan worked mostly on the RNN, and Jack worked mostly on the Additional goals and the visualization. After each step in the process, everything was discussed to make sure that every member understood every part of the project. The report was worked on by every member.
- Packages used for the project: NumPy, PyTorch, MatPlotLib, WordCloud
- Piazza Post: https://piazza.com/class/lprgcbeelor41p/post/563
- Code Links:
  - HMM: https://colab.research.google.com/drive/1O3YQOSZCaqxVQSEDagvwtD1qBbidZnFw?usp=sharing
  - RNN: https://colab.research.google.com/drive/1shYTg7Ls0KgRViTCv28gQIn8KE1qQ-0K?usp=sharing

2 Pre-Processing

HMM Preprocessing

For the preprocessing of the HMM model the dataset is tokenized as words, so each token is a single word. That way the model predicts the next word based on the current state, one word at a time. This was done in order to keep the same language that Shakespeare uses (his vocabulary). We decided to split sequences into each line as in Shakespeare each line tends to stand it's own, so generating coherent lines seemed the most important to us. Words with apostrophes and hyphens were tokenized as single words (not bigrams), and this was done in order to maintain the number of syllables in a word with an apostrophe. For example, "it's" is 1 syllable whereas "it is" is 2 syllables so the original apostrophe was kept. We also had to keep the hyphens and apostrophes because the amount of syllables in pronouncing Shakespeare is different from modern English, so we relied on the syllable dictionary given to determine the number of syllables to give a word, and if we were to split up hyphenated words we would not have the syllable dictionary to tell us the length of those split words (this was decided on all analysis of the syllable dict dataset). All words were parsed to be lower case, then when the poem is to be generated the first word of each line would be capitalized (this seemed the best way to parse all the words and data equally). For punctuation, all punctuation would be removed in the training, then re-added after the lines had been generated in order to create a final sonnet. After trying an initial model with just a word to syllable dictionary an additional dictionary was added to store the syllable count if the word was the last word of the line. This allowed the generate emission function to produce lines of exactly 10 syllables each time, which was a step-up from our original implementation, which could be inexact in the number of syllables it produced (based on the Shakespearean syllable counting). At one point we tried estimating the number of syllables in words which were not stored in the dictionary, and we decided to revert this feature as we wanted an exact number of syllables for each line. We also used the string package to clean up the input data and remove punctuation (except hyphens and apostrophes).

RNN Preprocessing

The preprocessing for the Recurrent Neural Network was slightly different than the previous method because it was a character-based model as opposed to a word based model. In other words, the RNN learned what the next character would be based on the previous seen characters as opposed to the HMM which learned what the next word would be based on the previous seen words. Similar to the previous method, we iterated through the file line by line and removed any commas, apostrophes, or numbers which would prevent our model from learning truly based on the characters. The processed file was then written to a new file called 'processed_shakespeare.txt'. From there, we created a dictionary where each key is an integer, and the corresponding value is a character from the text. This mapping allows us to convert integer indices back to characters. Similarly, we also created a reverse mapping where each key is a character from the text and the corresponding value is the unique integer assigned to that character. This allows us to convert characters back to their integer representation. Finally, we also initialized a NumPy array that represents the entire test string as a sequence of integer values which was later used for our input to the RNN.

3 Unsupervised Learning

We didn't use any additional packages for the HMM, just the implementation from HW6. To decide the number of hidden states we ran the model on varying hidden states and compared the outputs. It was clear that 18-22 hidden states produced the most coherent model, as with a lower number of states the word generation appeared completely random and for the larger amount of states there were too many consecutive words without a subject. Here are the results for the testing on the number of hiddent states (ran on the poem generation discussed in part 4).

For 10 hidden states, poem generated:
His height your farther still shall themes in i,
Mutual which my reason to friend prove seek,
Love winter the waste book world and o thy,
Which beauty so this fast best his woe her,
Which what i self-love move sweet all receive,
See the judgment the show are looks five men,
I tell say writ name me haste in i sing,
Own hate love nor nights i do state say thee,
Eclipses single is reign on so that,
So me love twenty when my in in would,
All ripe once your is babe my that thee not,
Could loss spring year of of rich of his which.
Than that am same of thought spring of wondrous,
With waste i thee i i lover is his.

For 14 hidden states, poem generated:
Gluttoning to thee me meet her her and,
Are them me and is my to each a and,
I with of day am should me night rich with,
Nor weep of doth his sovereign must love mother,
Then thou all i whom have i more to should,
Cast should self sweet put fair when my wrong shop,
Needs summer twain my simplicity wights,
Frame but yet else eye into stand true mine,
This mine song hath still pen image enjoy,
Which i'll dwells constancy and better in,
Confounds found let growing love give me to,
Death all it set gazed i their a like love.
That thou i which pursuit longer of my,
And dost cost how thou mine shall what that else.

For 18 hidden states, poem generated:
Affable a sure wilt happies should first,
Is cover show nor the poisoned in mine,
By death's have even dearest youth and right,
Of limbecks fiery show from did have it,
I could know outright hands sigh give lead on,
Prophecies the why of mansion eyes be,
Thou taste lasting my false morrow that in,
With are white than picture the my which her,
New others on him from the will no it,
For hell would lies believe hearing by the,
Let with ill love with will say disgrace gladly,
Sunken account i speed and like to i.
Grief to his learn grow'st me but beauty doth,
Which mend black much hath call summer's write look.

For 22 hidden states, poem generated:
Bright confined strange cannot not still long their,
'not thrust sweets i vermilion for sight other,
Victors of my thou bring answers deserved,
An numbers old can is dust that twain hear,
That verse ill-used the grows flies root now your,
Of own beauties alchemy put thee hate,
Are that gazing well to thy yet i a,
Worth to his thou dead of on past show march,
Hue cherish eyes the greater without my,
Of time worth love's heart hence to robbery thou,
I the the that i in lovely dost thou,
Pity the accessary do my doth.
By but so crime is glad youth fair as you,
A torn of necessary woman to.

For 26 hidden states, poem generated:
Draw on so strongly nor and own they more,
Ah blooms a seal amis say thee use love,
Poison lease thou will brow express salve found,
Delight to that a lay such honouring,
See my verses she new cloud me though o'er,
Then my ripe sad sight thy remain i thence,
What say thrall five by deceased faults with,
Of that hence it was this tell ill to to,

Making but kind of the contents thou due,
Nature's eyes being and in style time wert,
Night inward our by you heart but keep, You never cast pleasing thee doth true self.
Eyes me our in you your glory be,
Leese for from but self my still love i with.

For 30 hidden states, poem generated:
Spheres faith thy help alone i false and base,
The spite born may fear thy first counting sinks,
Mine particulars confess that these out,
His treasure of of gait of pyramids,
And heavy was gentle love outbraves in,
Vilest liker that proud unknown rosy,
A curse of did part obsequious from worth,
Put'st on memory large remote desire,
Which victors ill greater still by call they,
Swears being supposing thine send'st the verse,
Precious were in me mine think heart to shall,
Thee true tired bait his gust because he.
Must triumph love receive darkening she whose,
Have outworn thy pleasure cross i old clay.

4 Poetry Generation, Part 1: Hidden Markov Models

Code: https://colab.research.google.com/drive/1O3YQOSZCaqxVQSEDagvwtD1qBbidZnFw?usp=sharing

The algorithm for generating a poem has two main parts. First the generate emission function which generates each line, then the generate poem function which puts emissions together into a sonnet. The emission functions works by first choosing a random state then choosing the next word and next state based on that state and so on until there are 10 syllables. For each word the syllable count is checked and if the word puts the syllable count over ten then another word is chosen, until a word satisfies the criteria. The end syllables of each word is also checked, in other words if a word has a differing number of syllables when it is at the end of a line, then it counts as those number of syllables when it is the last word. This algorithm builds each line and returns 14 lines. Then the generate poem takes these 14 of those lines and formats them to the style of Shakespeare. The first word in each line is capitalized, the first 11 lines end in a comma, the 12th and 14th in a period, and the last 2 lines are indented. Here is an example of a poem generated

Example Sonnet:
With the self away with open seldom,
Will of willingly than patience may you,
Love and greater they to advised none it,
Her rich to convert of those convert shall,
But crooked fearless graces state i will,
Show wilt such and which thou my most are as,
How affords a dull of a pride ensample,
Fairer none by her hours part with your,
More deriv'd to for mind infant's this thing,
Thy glorious be for could by true stars thou,
That ye in canker bankrupt that vanished,
Age all away silence nor cheeks of truth.
Sacrifice now whom happy ever to,
This feet i attain of live let see stay.

The syllable count is near perfect in our poems as we track the end syllable count and we ensure that each line is exactly 10 syllables before adding it. The rhyme and rhythm have no code dedicated to them so they are subpar. As discussed in Section 3, the number of hidden states drastically affected the quality of the poem, too little states and the poem was too simple and too many states and the poem no longer had any meaning. We found the sweet spot to be around 20 states. Individual lines of the poem can make some sense, and are up for interpretation in the same way Shakespeare's original sonnets do, but line by line the sonnet changes subject. This is because we do not maintain context line by line and each line is generated independently from one another. In our opinion our poems retains Shakespeare's original voice as the sentence structure and vocabulary in the poems is very similar to what Shakespeare could have written in his own sonnets. This makes sense as an HMM is able to generate similar structures to the data it is given.

5 Poetry Generation, Part 2: Recurrent Neural Networks

Code: rnn_miniproject3_butters.ipynb

We used one-hot encoding to to transform numerical data into a binary matrix where only one bit in each row is set to 1, with the position of the 1 corresponding to the character's integer mapping. This representation allows the network to handle categorical input data like characters in a sonnet. We defined the RNN class such that it includes an LSTM layer to process the sequence data with 200 hidden units, a fully connected layer to map the LSTM outputs, and a LogSoftmax layer to convert the outputs into probabilities for each character in the vocabulary. The LSTM's hidden states are managed across sequences for retaining the learnt information through time. We then sliced the input data into batches for training to ensures that each batch is a sequence of a specified length and creates corresponding target sequences for each input sequence. We used a batch size of 128 and a sequence length of 40 characers. We also incorporated temperature scaling to the output probabilities, affecting the randomness of the prediction. Lower temperatures make the model more confident (but also more conservative), while higher temperatures result in more diversity (but also more mistakes). We tested [0.25, 0.75, 1.0, 1.5] as our temperature hyperparameters. Our results are as follows:

Temp = 0.25:
shall i compare thee to a summer's day?
To time that were be not mand the bless the world to tine to stare
To be this start on this stine to more's light worthing to me
To be note and thy spandes thy strong to mich
With withored bus and once to my best bad with
That storn of thy stranse and all that mines
Then be to most wound ant theneming on stones of mend
So fair and bared of beaute tembsed
Than thin well were that stain we proancess are do black
Or a prave impesides of bear
As ara now stoll and thou dost time do mend:
And my fase

Temp = 0.75:
shall i compare thee to a summer's day?
Be where tare boughis for thoughs of my spease
Wheres all thy pand of tought that mear
Whene this shalded bud men in time
And me tanked of thou my dead than time ways beart though a prigest it alond
When that be thy sent in winker as my sang
That moured a tompance can me deeping

Wher bull wond all wonke thou most thee wimb abjaind
As alose befors of tented and all thee
That but suck the from that beauted:
Buc the wilt born thy serfors and be not
So that which true tanker all thy with's senfers

Temp = 1.0:
shall i compare thee to a summer's day?
When shalls that store'stered barn that sweet stain
And than thy pling mind the wants wold are beauting
Thare the stalter in thain are madress with hasse of bestedse
Then than serves if my beause of thim thou seem
And that by might in thy parte is me but make Than all the wints trow wile by this sweet with move time me dain to by thin well
As are wouth thie the summant wear would wit
That shing on the wournone to mich
And be where brentered thou my brain
The walls to me desaiced bo thate
In thy

Temp = 1.5:
shall i compare thee to a summer's day?
And trave and thy love that woll be would the fart alove be
That when In serisent that time
And strangly and maye some no heart to me
And siched and are thines and this flame
And beauty that ser that but my lay nim strences than that tays time

O hin wee should with the sweets aloning own beaution
Betune alledsece all suction and summer winct of mase me dellow
Whick of thy pant in thy beauty soon thou and have
That may that temmine that thy wancest be
To mine eye of may al then make thy paid
Th

From our results, we see that the lower temperature results in sonnets that have complete words while higher temperature results show incomplete/nonsensical words. Regardless, the RNN network was able to generate text that was similar in style to Shakespeare's sonnets. That being said, it was not able to maintain any rhyme, meter, or structure. Additionally, no poem made any grammatical or logical sense.

6 Additional Goals

Code: https://colab.research.google.com/drive/1O3YQOSZCaqxVQSEDagvwtD1qBbidZnFw?usp=sharing

Rhyming

For our primary additional goal, we implemented rhyming into our poem generation. Since the sonnet follows strict rhyming patterns, we were able to figure out what rhymes Shakespeare uses by looking at the last words of rhyming line pairs. This required preprocessing the data in a slightly different way (seen in the rhyming_preprocesssing function). Essentially, we took the rhyming pairs of every sonnet and put them into a dictionary (the rhyme_map as we refer to it in the code). Each entry was a word key with a value of a list of all of the words that were used as rhymes in the sonnets. The dictionary went both directions (ex. hat would be in the cat key and cat would be in the hat key). To generate our new rhyming poem, we created a new emission function (generate_emission_rhyme) that required a list of rhyming words as an input. Since it would be difficult to ensure each line was the correct number of syllables when placing the selected rhyming words at the end, we performed the HMM generation as if the rhyming words were the first word, and then reversed the output emission at the end. This produced relatively good results, and it seems that working backwards does not noticeably diminish the quality or creativity of the poem. To generate the rhyming words we created the function generate_rhyming_words, which uses the rhyming map created to randomly select rhyming pairs and place them in the correct order for the emission generation. Since the words are randomly selected, we do believe there is room for improvement in the selection of the words that could result in more coherent outputs. Also, during the generation of the emission, the starting state following the selected rhyming word is completely random. This could possibly be improved by starting at an estimated state for the rhyming word, which may improve the quality of the poems created. Overall, the implementation of rhyming seemed very helpful in making the poems resemble the Shakespearean sonnets the model was trained on. Although not always more comprehensible, the outputs at least better fit the correct format of a sonnet. Below we have included two example generations of our rhyming sonnets:

Sonnet 1:
Sits do not still devil grave keen stands longer,
On in my alone still thy can me hence,
Their heart my may good the from her much stronger,
Do mansion is speed in against defence,
Did from world's grows are exceeds though truth yore,
Or store with wilt doth issue of which gaol,
That age i often be may but that store,
Motion to are thy maturity bail,
Wretched do as will am tender in owe,
Cut loved have will to to-morrow and pry,
Enlighten define for his on bent show,
Her as dimmed mind none ever jealousy.
For fiery white dear have and the eyes frown,
A thy love thine did my you like yet down.

Sonnet 2:
It him in with loss me is this for breast,
Of more idol his all that plain misplaced, State a of yours quicker have yet expressed, Be and false invent sovereign thou disgraced,
Him night wound wonts deprive in and anon,
And all than ever smell love my of fiend,
Yet be at minds you white of fool alone,
Heir ought budding thine thee is trusting friend,
Chose fragrant canst what cheer and honor transferred,
One precious in love but add line no spread,
Survey more durefull down thou is and erred,
Spoils double prevailed woe dry buried.
If crooked returned love the weigh her boast,
Follow faults in dost but being poor ghost.

Other Poetic Forms (Haiku)

For our second additional goal, we tested the ability of our model to generate Haikus, which are three line poems that follow an ABA rhyming scheme and have 5, 7, and 5 syllables in each line, respectively. This required us to create another emission function within our HMM class, generate_emission_haiku. It functioned in a very similar manner to the generate_emission_rhyme (also taking a list of rhyme words as input), but ensured that the first and third lines were 5 syllables, while the second line was 7 syllables. For generating the rhyme words input, we created generate_rhyming_words_haiku, again using the previously created rhyming map. This function randomly selected a rhyming pair for the first and third lines (from the rhyming map), and selected a completely random word (not necessarily in the rhyming map) for the second line (since this word did not need to rhyme with another). We also created a separate generate poem function (generate_haiku) to format the poem correctly. The result was fairly pleasing, although we believe the shortened nature of the haiku make it a bit difficult to generate something sensible. However, we also know that many haikus tend to be a bit difficult to understand, so in a way it almost made the output seem more realistic. Below we have included two example generations of our haikus:

Haiku 1:
Live stops mistress pride
Thou my thoughts sigh that winter
How barren grow hide.

Haiku 2:
Eye's makes you my crime
For heat discloses beggar
My dazed on not time.

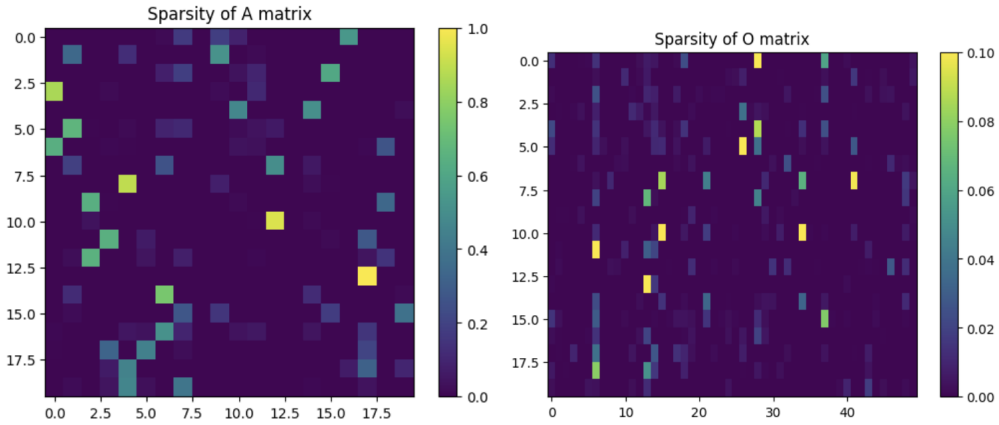
Incorporating Additional Texts

For our final additional goal, we incorporated an additional text, Spenser's Amoretti, into the training of our HMM. This was done by simply processing the Spenser and Shakespeare datasets as one combined text. We then trained the unsupervised HMM on the output obs and obs_map from the combined text. The difference in the sonnet is certainly noticeable. The outputs appear to be more varied in the vocabulary used. However, they also do not make much sense, which could be a byproduct of combining two different styles of sonnet writing. Below we have included two example sonnet generations using the combined dataset:

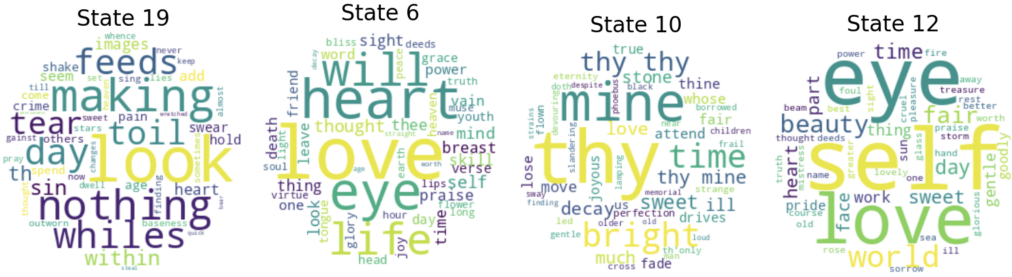
Sonnet 1:
And thine on from in then chips compile plot,
Ever gone fair we i is subtleties,
To any shames slavery down else for not,
Thou should men doth enforced lost tempteth lies,
With see do aspire i self carcanet,
The none so comfort steel books the confounds,
Keep spell name a i go sweet o birth set,
Move pride stay by he were most dearer sounds,
Proud as toil and of shall scope lie prevailed,
Their see and thy of from you me while knit,
Love my in most monument so assailed,
Therewith misdeem air play every wit.
Knowing that your morn publish do find alone,
Different proud pleasure glass cruel anon.

Sonnet 2:
Eyes praise masked the 'tis my bearing self brood,
Will thou blushing alive i well this wit,
Which at eyes time view hell her if but blood,
Pain sportive of that of if for thee writ,
Straight dear gentle me of self beside spread,
Mortal love worser for or gravity,
In above which felly have no buried,
More my it scorneth not of store are eye,
Whence my will be golden tempests and now,
In and such but peace art tibey thou bunch hits,
The fair must season spite loves and home how,
Man they knows unused yet pity fits.
I wherein quicker me doth in those laws,
Freewill thoughts night amazed majesty cause.

7 Visualization & Interpretation



Similar to the visualizations created in Set 6, we generated sparsity of transition and observation matrices. From these visualizations above, it is evident that the transition and observation matrices are both fairly sparse. The A (transition) matrix's sparsity implies that there are only a few states into which the HMM model can transition. The few non-zero values represent the strong probability of transitioning between certain states. For example, it's apparent that transitioning from state 10 can only result in state 12, as evidenced by the single yellow square in an otherwise dark purple row in the Sparsity of A matrix. This aligns well with the word clouds below, as state 10 is predominantly associated with possessive and adjective words (mine, thy, thine, bright, sweet, etc.), while state 12's words are primarily nouns (eye, beauty, love, world, self). It makes sense that an adjective or possessive word would be followed by a noun, so it would this transition would be expected to have a high probability. The O (observation) matrix's sparsity indicates that there are only a few observations likely to arise from a given state. In this case, the few non-zero values represent the strong probability of emitting certain observations. This is logical, as these would be words that are used frequently through Shakespeare's sonnets.



We also created word clouds for each state to demonstrate the most common words of each hidden state. Above are four of the word clouds and below are lists of the prominent or intriguing words from each of the states. At the end of each state's word list, we identified shared features within the grouping. All 20 of the states can be viewed at the end of the linked code.

Prominent and intriguing words for four hidden states:

State 6: love, heart, eye, life, will, thought, lips, tongue, breast, head → Words associated with love and body parts State 10: thy, mine, bright, sweet, joyous, ill, fair, thine, much → Possessive and adjective words State 12: eye, self, love, face, beauty, time, day, heart, world → Nouns State 19: feeds, making, tear, toil, look, nothing, whiles, sin, images → Verbs

As shown by the word clouds, each state tends to group words that relate together. There are also some words that are repeated throughout many states. These are primarily words that appear in the sonnets often.