Kelsi Riley & Sakthi Vetrivel
Machine Learning & Data Mining
Miniproject 3

Caltech Earthquakes
Caltech CS/CNS/EE 155
March 12^{th}, 2018

---

1 Introduction

- **Group members**
  Kelsi Riley
  Sakthi Vetrivel
- **Team name**
  Caltech Earthquakes
- **Division of labour**
  Kelsi implemented the HMM, and focused on improving it. She also did the preprocessing for the HMM, and the unsupervised algorithm. Sakthi implemented the RNN and worked on improvements for it, as well and the visualization and interpretation of the HMM.

2 Pre-Processing

For our pre-processing of Shakespeare's sonnets, we started by converting each poem into a list, where the list contains the words of the poem in order of appearance, with a newline character between lines of the poem. Then because our HMM uses integer representations of emissions, we created two dictionaries: one mapping words to unique integer values and another mapping integers to unique words. This initial preprocessing was helpful, but when we looked at the values stored in our dictionaries, we realized that capitalized and lowercase words were considered different emissions with this processing, and if a word was followed by punctuation, it was considered a different emission than if it was not.

This prompted us to make each word lower case as we processed it, to ensure that words are considered the same emission regardless of their capitalization. We also decided to remove the characters '.', ',', ':', ';', '(', ')' and '?' from the poems we processed so that we didn't have many different emissions associated with the same word. We decided to leave in the apostrophe due to its presence in contractions, and actually being a part of the representation of the word. This updated preprocessing was an improvement from our initial attempt, but ultimately fell short of what we needed due to the presence of phrases in quotations in some of the poems (i.e. 'Will' in sonnets 135 and 136 and 'not you' and 'I hate' in sonnet 145). This lead us to need to adopt a somewhat more complicated process for removing quotations around words.

We ultimately ended up deciding to remove an apostrophe if it was in the beginning of a word and the word was not a word we recognized as beginning with an apostrophe. Then, we would remove the next apostrophe in the line. This allowed us to remove the quotations around certain words and phrases in the poems.

Because sonnets 126 and 99 adhered to the structure of a sonnet much less rigidly then the rest of Shakespeare's works, we decided not to train our models on them.

Finally, we wanted to be able to generate sonnets that rhyme, so we ended up extending our preprocessing to enable doing so. To do this, we used our knowledge of the rhyme scheme of Shakespearian sonnets: abab cdcd efef gg. Then, as we processed our poems, we added the unique integer representations of last words of lines that rhyme to a rhyming dictionary. This rhyming dictionary mapped unique integer representations of words to a list of the integer representations of words that it rhymes with. We considered altering this so that if two words rhyme with each other, then every word that rhymes with either of them

will rhyme with both of them, but ultimately decided against this implementation due to Shakespeare's liberal use of slant rhymes. We thought that generalizing rhyming this way could possibly produce "rhymes" that hardly resemble rhymes at all.

3 Unsupervised Learning

For our unsupervised learning, we used the Homework 6 Solution Set. We ultimately ended up choosing the number of hidden states due to computational limitations.

4 Poetry Generation, Part 1: Hidden Markov Models

14 Hidden states, 200 iterations: me what my old bear it either's veil are song translated virtue to thine wont be when the earthly lays that but a thee care is dissuade in their numbers forget i live teeth which else that be leisure when prove own twain alone of still mock love forgot flame my ill taught sometime they then but love with things unrest where summer as still not thither out with to time i thou thy bear eyes to thee lost an therefore bold repair water therefore must is tyranny were see be augurs but me a left looks fair love and his more so eclipse is thing be concord where dully are respect is i

of then vile loving earth then lends how call speak that short in him within publish is me the far say in of hardest the all why i in civil art a time's eye this shadow of be he which love's of doth lie sense whence wild that but other use remove the hide the of rose forwards mine sense why sorrows fame with lame or flowers the love thereby deformed'st can antiquity rare of my heart be justify as thou thine being give shall above so might and am are are that i it to for his parts be mine think o sworn is shifting pity my sight voice so lines from when i effect none night

5 Poetry Generation, Part 2: Recurrent Neural Networks

Initial Implementation

For our preprocessing for the RNN, we decided to first break down the text file by line, and for each line we parsed, we made sure the line was not empty (not just an endline character) and removed any special characters from it. For example, we wanted to "Hello!" and "hello" to be processed as the same sequence of characters. We then finished each line with an endline character and added it to a accumulating string, which held the contents of the processed text file. After processing the input, we created dictionaries to convert each character found in the processed text to an integer, and a dictionary that converted integers to characters. We then generated a data set splitting this string into sequences of 40 consecutive characters, and converting the sequence of characters into a sequence of integers, and used the 41st character of the sequence as the y value, again, after converting it to an int.

We implemented a recurrent neural network using the Keras package for Python3. Using a sequential model, we had two dense LSTM layers of size 200, and one output layer. We calculated our loss using categorical cross-entropy loss, and 'adam' optimization. We also fine tuned our batch size, testing values of 32, 64, and 128. By observing the poems produced for these batch sized after 10 epochs of training, we settled on a batch size of 64. We also fine-tuned the size of the LSTM layers, trying sizes of 100, 150, and 200, as suggest in the project details, and settled of a size of 200. We then trained this model for 125 epochs,

converging to a loss of 0.4146 from a loss of 3.45 after the first epoch. To improve our model, we also looked at more complex RNN, using two dropout layers between the LSTM layers, and fine tuned the dropout probability to eventually choose 0.4. Initially, the model was only trained for 20 epochs, but we found that the loss still hadn't converged so we continued training until the loss did not improve for 2 epochs. We used a window size of 40 as the instruction suggested, but later moved to smaller window size of 25 in our attempts to improve our model.

To generate our poems, we used our seed sequence "Shall I compare thee to a summer's day?" and processed it, and for a sliding window of 40 characters, predicted the next letter in the sequence. Our model gives us an array of probabilities for the next character, given the previous 40 characters, so using this array of probabilities, we sample from the population of characters accordingly for a given diversity value. We also counted the number of newline characters that were predicted in the entire generation process, stopping after 14 newline characters, giving us 14 lines in the poem.

Despite training for so many iterations, our LSTM did not successfully learn sentence structure or sonnet structure, as we see in the poems below, few of the words produced are real words. However, for the brief segments of the poems that contain real words, it seems to follow some loose sentence structure. For example: "i love you" and "of the braid". As a result, the poem quality seems fairly low, and this is a direct result of the character-based nature of the neural network resulting in jumbled English. Because there are so many more sequences of characters than sequences of words, the RNN takes more training data to train, but both sets of training data were still generated from the same text file. It took much longer to train the RNN than to train the HMM, again, because it starts from a lower level of understanding, since we are working with characters instead of words.

Poems

—— diversity: 1.5
—— Generating with seed: " shall i compare thee to a summers day
"
shall i compare thee to a summers day
weat kerp tellv in these i would to hath
my self iil iiddsu your thlltt my self brane ereed
but ceatt that weal i love you be teildde
and me altereasonse of the braid and lind
which treals mind eye is is a lawvereo
at tenmn lines bety paming iimeshsy
or at dolg pdr tiat weadddleditg tiink
eor shamls iampeut sp will he's shcd might
sevoy seep tidu thou 'liserpeas nor
j thak oyck tine suelt so lem so ku haln
and nock i tas of fold cach or pattry
but be thy liate me thrnugh mights me sn botn
and in holaskeri hrln with the trwe doth green
and confoundane farth in thee tie live"

—— diversity: 0.75
—— Generating with seed: " shall i compare thee to a summers day
"
shall i compare thee to a summers day
when that wilt nottncry that fell asd feidt
that it 'bol gor moctatd i do dispilts
and diary my self i'ck tren to the most
but when your changent of this weil
byt sickt his tputatt mot i loow more eeee
to say toe borcmest whereup the bear
thy presclv ceauies ifart he lile artire
for whose winter's enoling on the rulp're
when sesimg a betteiry ocrure of thy deeds
therefo thy putlok dead trealed thou art
o what a worthsed wron delive no mane sehmnts
oo aly of these falsehe move's fresh ceserity
then the means me with vinter did stansed
and dotnt and in habkt and it gaults light"

—— diversity: 0.25
—— Generating with seed: " shall i compare thee to a summers day
"
shall i compare thee to a summers day
aid uosthfr this wirte doth beauty stail
thou mayst be thy oudsent'st a linit sade
but when your count in these cannot chind
o carve norer mine him though mews the even
but day doth daily draw my sorrows line
so thy freat gift woon be forgouingnl
for higheo of line ow well my heart deegines
so fotth the blow of with dupy steet selbit
ald my hoade fyen siln liss lysbs'bd and were orisit
oatt reason haved the stard or thy sweet graces
beauteous all fellls tine world have erreemed
more that my self but was donf iis oun
gow many lambs kild and they acvodance seegng
and all the dead no nore drtbl dole"

6 Additional Goals

Haikus

We adapted our RNN to compose haikus. To do this, we created an file of haikus, where each line in a haiku in separated by a tab character, and each haiku is separated by a newline character. We also changed our window size from 40 characters to 15, since haikus are much smaller, and a 40 character seed would make up a significant portion of the poem. We continued to use LSTM layers of size 200. We then trained the RNN on this data set for about 100 epochs with a batch size of 64 (the batch size we found was effective for our sonnets). To generate our haikus, we used the same skeleton as the sonnets, just adjusting the window size. The results for the haikus were much more promising than the sonnets, which was surprising considering the training set was smaller. Here are some of our favorite haikus that were generated:

light snowy breath to read
old tombstones hot wind shadows
the roadrunner's beak

furled umbrella
turns into a cane
chernobyl vells

Of course, many of the haikus we generated often contained just jumbled letters, and when they did contain words, they rarely made sense, much like the haikus above. However, the model was able to predict syllable count fairly accurately, as you can see the first haiku follows the 5-7-5 pattern, and the second haiku is close. Given more time, we would have liked to experiment with the Word2Vec package and worked to predict words that were similar to the previous words in the sequence.

More Complex RNNs

To improve our RNN, we decided to implement a more complex model. Instead of just using two LSTM layers of size 200, we decided to include dropout layers as well to avoid overfitting.We also used a smaller window size of 25. Thus our final RNN model consisted of two alternating LSTM layers of size 512, alternating with dropout layers. For these dropout layers, we experimented with probabilities of 0.1, 0.2,, 0.4, and 0.5, settling on 0.2. Moreover, instead of predicting the character following a sequence, we trained the model to predict the character preceding the sequence. This was in preparation for us to generate rhyming poems.

Given the chance to do this again, we would have liked to also train with a validation set, and have stopped training when the validation loss stopped improving, instead of depending of the training loss. Here are the poems generated with this approach:

—— Generating with end: ue by the grave and thee
—— Diversity: 0.25

and in this cold complaining

she would have trained crie
and in his and he like again
my art thou art in me must roves to side thy state
and in your offences fived in your devory
of fathered bester and with dinkilling eye
that she more living so offence alove
in either kill of beauty tend
and with a concestion makes her wind
and with your shame to supretise is scapening on to love my love
that my after for my days dad make the mettle wit
against my valthough it should nor gistless
cinfounded in a mined eyes
the tongue by the grave and thee

—— Diversity: 0.75

but loves her painted hend
with whose what eyes now love to my
and that can men shall heavenly for see only most to shows and toney
to make the mudder's grace
for thy sweet to time's eyes to hast
but no so is my love to cay
and to the humbers at the spirit on thee
'didst thou the sefle but in his rain
to from the best this true for then the eye alone
even in my love com from that at the reason
to lives my sorrow nature had't teou poy
to fair thy loses she lacks and unneed the bound to make their sidness
fit and the trembling eyes
the tongue by the grave and thee

—— Diversity: 1.5

i should in cartle's dame
was fined in lidob giden kies
then boldness her beauteous i will
hinour to elmantinal condensmaking
with those and burd nor let
to prubify to leave th from it
when shalt thou my have be end
so do am state at living eye and heart it may be gone lix
the other hours are so co civitate
the colous o'erds to hell for love and bift from this all treed
and when that gave to trespect thy state

but were the had taught the cloud or all apay
and in her hourffece baskys-are tongue by the grave and thee

Larger Training Set

In order to give the RNN more information to train on, we decided to include another file along with the spencer file given to us. Finding other sonnets composed by Shakespeare, we decided to process that text as well and include it to train our model. This larger training set was used in our final RNN model.

Rhyming

For our rhyming using the RNN, we used this implementation of the more complex RNN. Our first step was to generate a rhyming dictionary using the CMU dictionary and parsing each line of the sonnets. Using the CMU dictionary, we generated the pronunciations of each of the words in the Shakespearean texts, and compared the pronunciations of the ending of each of the words, so each word mapped to the pronunciation of its ending, and each pronunciation mapped to a list of words with that ending.

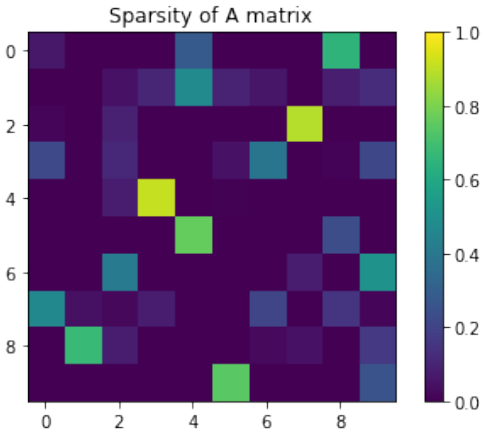
Now to generate the rhyming poem, we find the endings of each line and generate the preceding characters based on the word. First, we find 7 random words in the sonnets, and make these the ends of lines 1, 2, 5, 6, 9, 10, and 13. Then, we use our rhyming dictionary to choose a word that rhymes with the line endings of the aforementioned lines and make these words the endings of lines 3, 4, 7, 8, 11, 12, and 14, such that we have a $ababcdcdefefgg$ rhyme scheme.

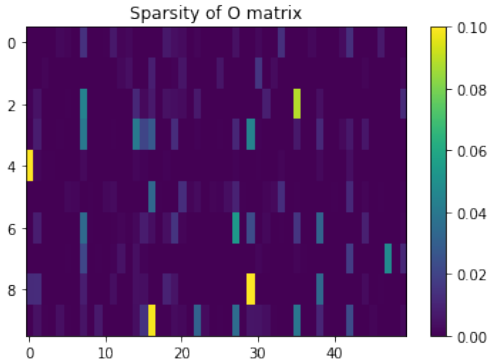
Below is one of the poems generated through this process:

Have women witagainst the cause but in the horn,
The sparm'dall wither yet love me all contented stair,
To beand croused could back their mildress' sky forlorn,
Effects and dischance andmore beauty of thyself all stair.
But his smokest love some is and now appear,
With that praise for living chance or thy burn,
Friendand for thee and steared berience more you peer,
Lively and confounds his artand heavenly one can spurn.
That true whan attening of her lind and horse,
Thas thoosebut if the would be maunt the heart,
Of all love's sout'st she dings with thy scarce,
Eyesand never was so watch the for both's transport.
To the priseand sily burupt make tears they sphere,
Thoutment on that gave darefrown from the dangerous year.

7 Visualization and Interpretation

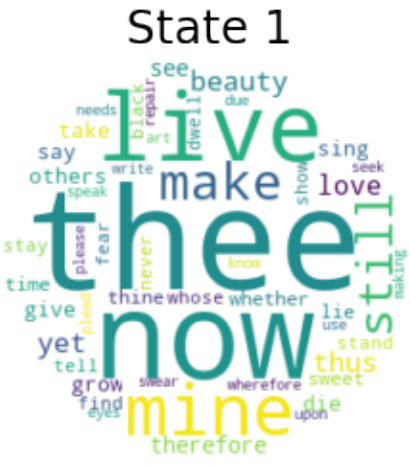
Visualization of the sparsities of A and O

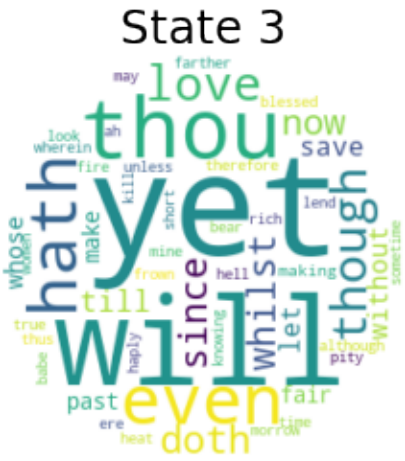




From the visualization of the sparsities of these matrices, we see that most of the elements are zero or very close to zero. In the visualization of the sparsity of matrix A, we see that this means that for any given state, there are only a few other states that HMM will transition to. Likewise, for the O matrix, this means that for each states, there are only a few observations that are likely to be emitted.

Visualization of states





For State 0, we see that the top 10 words associated with this hidden state are eye, doth, may, say, will, must, far, love, seem, and look. Amongst this group, it seems that majority of the words are verbs and other words that might follow a noun.

For State 1, we see that the top 10 words associated with this hidden state are thee, live, now, make, mine, still, love, whether, beauty, and see. This group is mainly made of adjectives or verbs that would generally precede a noun.

For State 2, we see that the top 10 words associated with this hidden state are thy, love, one, mine, thine, full, made, three, speak, and th'. Similar to State 1, this group of words seems to be made of adjectives using to describe a noun or follow a verb.

For State 3, we see that the top 10 words associated with this hidden state are yet, will, hath, thou, since, love, though, even, now, and save. This group of words suggests that this state represents the beginning of

prepositional phrase, and words that might follow a noun in the predicate of a sentence.

For State 4, we see that the top 10 words associated with this hidden state are thee, love, art, true, seen, old, decay, heart, death, and will. This group of words represents nouns that are frequently used by Shakespeare and are words that frequently follow verbs or begin a phrase.

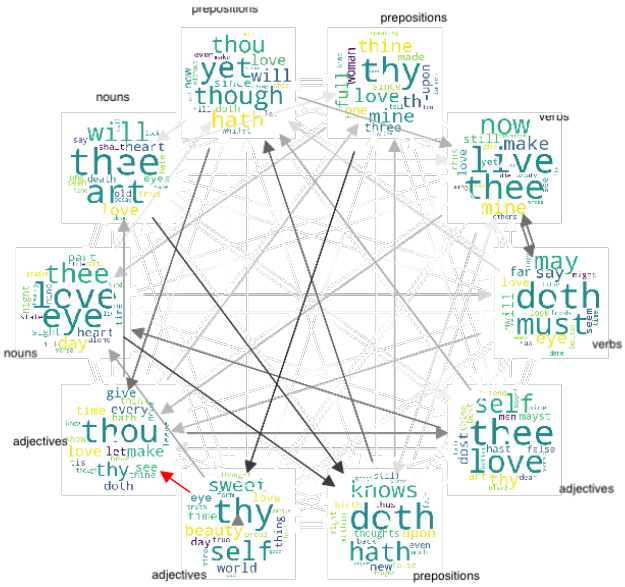


Figure 1: Graph of transitions between different states in Hidden Markov Model

This graph shows that transitions that we predicted while looking at the individual states are fairly common. We see that the states tend to represent certain parts of speech, and as a result, transition probabilities between certain states reflect the probability of certain parts of speech following one another. For example, we see that the State 4, that we considered the "noun" state often transitions to State 0, which we considered the "verb" state.

From these visualizations, it seems that as HMM learns from the training data, it uses the hidden states to determine different parts of speech, and the transitions are used to determine which parts of speech the model should transition between to mimic the Shakespearean language used in sonnets. With more hidden states, these states would likely extend to include parts of speech with more or less syllables as the machine attempts to account for the iambic pentameter nature of the sonnets.

The following sites were used for help with implementation:
https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/
https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/