---

0 Introduction

Group members: Po-Chih Chen, Wenzheng Liu

Team name: PCCWZL

Division of labor:

Po-Chih Chen: Implemented the HMM model. Compiled part of the final report.
Wenzheng Liu: Implemented the RNN model. Compiled part of the final report.

1 Pre-processing

HMM:

For our HMM model, we tokenize the data set in terms of words. That is, a single word is viewed as a token, and we do not keep some words tokenized as bigrams. Hyphenated words are kept as a single token in order to be consistent with the provided syllable dictionary.

Initially, we choose to use a poem as a singular sequence so that the transitions among lines and even stanzas can be learned. By contrast, we choose to use a line as a singular sequence for our improved HMM model described in Section "Additional Goals". The first reason is that it seems that the transitions among lines are not well learned even if we use a poem as a singular sequence. The second reason is that it is natural to use a line as a singular sequence if we want to introduce rhyme into our poems.

Finally, we found that the punctuation in all the poems includes !'(),-.:;? Among them, when training our HMM, we ignore !(),.:;? and only keep '- because these two can be part of a word.

RNN:

For our RNN model, we tokenize the data set in terms of characters. That is, a single character is viewed as a token, including the characters such as punctuation and '\n', etc. There are 61 different characters in total, which are one-hot encoded to train later.

Our training data consists of sequences of fixed length (40 characters) drawn from the sonnet corpus. Semi-redundant sequences (picking only sequences starting every second character) are used to make a tradeoff between training time and size of training data. To avoid the non-word generation, in Section "Additional Goals", we encode words into 80 length vector through embedding, not one-hot character encoding anymore. Although this advanced method can avoid the non-word generation, the size of training data would be much smaller than the initial method, which would easily cause overfitting.

As a special reminder, we encode the punctuation in our one-hot character encoding regime, while ignore them in our embedding word encoding regime, which means that we need to add punctuation manually.

2 Unsupervised Learning (HMM)

We implement our hidden Markov model based on our code from Homework 6, which implements the Baum-Welch algorithm. We make this choice because as suggested in the problem set, HMM packages for python tend to be poorly documented. We have also checked that our results in Homework 6 are correct based on a TA's Piazza post.

We train our HMM model with various values of number of hidden states, including 16, 32, and 64. Then, we determine which one is the best according to meaning, metric, and rhymes. According to the results, the model with 64 states looks the best, but the difference between the 32-state model and the 64-state model is not quite large. Thus, we do not try an even larger number of states as the gain in performance is not comparable to the increased training time and model complexity.

3 Poetry Generation, Part 1: Hidden Markov Models

Our algorithm for generating the 14-line sonnet:

1. Do the pre-processing and unsupervised learning as described in the previous sections.
2. Use the obtained model to probabilistically generate a sequence of emissions, i.e., a sequence of words, that is long enough for a poem, e.g., 140 words. The probabilistic generation process is the same as we did in Homework 6.
3. From the beginning of the generated sequence of words, we collect words one by one until the number of syllables sum up to at least 10. The collected sequence of words is the first line of our poem.
4. Continue the process until we obtain 14 lines.
5. Add punctuation manually. Specifically, we always use ",,,.,,,.,,,.," as the punctuation of the 14 lines.

This naive way to generating the 14-line sonnet in general does not give good results. A sample sonnet generated by the naive HMM model using 16 states:

What did should as that life right beauty than,
And hand and to fleece and that be the rest,
Of so in expiate not whereto all,
The hand that due dignity fell when although.
Thou his it works it in wonder bath merit,
Or beauty's why should why was the says them,
The no heinous possessing intermixed',
Friend' will your cannot my purposed for invocate.
Not not to grace for thy love's beauty despite,
Fresh in a conquest so acquaintance leave,
O first rising in dead when forgetfulness,
Or sovereign were within your time account.
Such the princes' in my self on by nature's,
With my love thy key or all thou from thy.

A sample sonnet generated by the naive HMM model using 64 states:

So silent lies eye part upon the rose,
Survive all contents what therefore to but,
Story curls hang past have not these loving,
Lies we bad vision wherein i a nor.
Let desire him live invited what thou,
New now for heart why his all spend'st in thy,
Far can it was mine own if your wond'ring,
Verse so stay melancholy when now wards.
For hymns no muse for a mark of soon to,

Grew carve hast fell dost did far beauty wood,
See when thy self to much purpose die how,
Dull my character cannot gaudy the.
Pleasure heir whose keep my self word spent thee,
I and league infant's makes not thy yet with.

There is often no rhyme. The generated poem also violates, at many positions, the rhythm, i.e., meter which Shakespearean sonnets follow. The syllable count is close to but not always 10. It can be sometimes a little large, like 11 or 12 when the last word of a line has multiple syllables. The meaning of the poem is not very clear, but it retains Shakespeares original voice to some extent as all the word come from Shakespeare's poems. The poem generated by the naive HMM model using 64 states seems a little bit better than that generated by the naive HMM model using 16 states qualitatively, but it is hard to compare them quantitatively as both poems are still far from Shakespeare's real poems.

4 Poetry Generation, Part 2: Recurrent Neural Networks

Our algorithm for generating the 14-line sonnet:

1. Do the pre-processing, which includes data cleaning (remove the number of sonnet), one-hot coding, 40-length character sequences generation, training data generation (learn first 39 characters then predict the last one).

2. The RNN model generated below:

```
# define model
model = Sequential()
model.add(LSTM(180, input_shape=(X.shape[1], X.shape[2])))
model.add(Dense(vocab_size))
model.add(Activation('softmax'))
# compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

   Here, we use a single layer of 180 LSTM units to ensure the sufficient training process. A standard fully-connected output layer with a softmax nonlinearity linked later. To generate poems, we draw softmax samples from our trained model. We use categorical cross-entropy as our optimization objective because the problem is a multi-classification problem.

3. Train the model with 200 epochs to enhance the accuracy.

4. We add the temperature parameter in our model to introduce randomness.

```
from keras.models import load_model
Model = load_model('model.h5')
temp=0.1
Model.pop()
Model.add(Lambda(lambda x: x/temp))
Model.add(Activation('softmax'))
print(Model.summary())
# compile model
Model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

   When we increase the temperature, the randomness will be amplified, which could help us against overfitting.

This one-hot character encoding RNN to generating the 14-line sonnet in general does not give expected results. The sonnet does learn sentence structure successfully, however, many non-words also exist in the content. A sample sonnet generated by the one-hot character encoding RNN model:

Shall i compare thee to a summer's day.
Not my miste my mind, wit stee my hall lie,
Which richang a perfumbence to thee,
And such a too me give mad in hamp:
Sweet beast belole thee for more who shouldss decar.
O' whow for you, blooks of me worldes time,
Look it self could true life to rege,
So shall thou dost part content which thy side,
In ate were bried more beluly of may.
'Ging a sar,
O' is coldel of earne, for my say
Now fair requite with thoughts when thou alone,
I do they I am thy self this self,
Of my dear love be assets my live,

Compared with the HMM above, we find that:

1. Meaning: Since HMM generates lines independently, the meaning between each line is unrelated. While the lines generated by RNN are correlated because of the adaptive learning process.

2. Words & Structure: Since HMM generates structure setting by people, using word as tokens, the words & Structure is much better than the sonnet generated by RNN.

3. Runing time: RNN need to consume more time than HMM in general.

Generate sonnet with different temperature:
Temp=1.5:

Shall i compare thee to a summer's day?
No look have will preperous this coor thin, and trats you gray foul,
Is thou be deat wime thee mayse me amise,
Or ad my one must is my bent im it,
My sinf of me, be kenmped of thy love.
Iver's fair ran moul's feare) O lires,
But the world thy best is able devil if chise,
But dear chepkraged my brart shall be give,
All thou ansurined the time (though I know.
But from thy rider why dost but be didgaing,
When so my thought hadrs your to cartal eys?
The clost to evenge and sicke vilt, and duppare hears
Dit strains of would that beloness mays, dit,
My bend I looking of his parts with lies,

Temp=0.75:

Shall i compare thee to a summer's day.
No long rubure, thy well beloness more.
Thou art comptance I amisow heaving lake.
Thou speingly be not remembers' pest,
That is my self thy sweon that beld I neade,
That Tigets dist meet, belied thee thus say,
Which chose firgulf oque all amove,
The heaver of this conting so more true sing lies,
So wit but neal my best alone delie,
For I in the parts beloved angred'g file.
Let no how her is being being cort,
Riengs dull buth for my dididid what well mine,
So oft be, me thou mayst thou thine of thee
Calls that with fell belomy doth dead world,

Temp=0.25:

Shall i compare thee to a summer's day.
Not my sake blooted weed hus sweet bell me.
So love stall can me heaven th timplates',
Why her I dayst may strright that love grace, anow.
Such like Enemeds' as I my mortung stay
That thou be dead tould their in his glunts grace,
Bring find threvon thy hear with a sum all,
To sind thy twould have I shows thy ersect,
I think on thee not so lov'st so forsad,
The wirns for morring that they dead tome,
Look deal lungge me which shall stringless keepst,
Thou art covet with heavy love you beftere.
Let that be the shows must bation made one,
Thou art conttant doth aw heaving and days,

Comparing these three sonnets generated using different temperature, we find that higher temperature will introduce more chaos, more creation and more innovation.

5 Additional Goals

Rhyme:

As we described in Section "Poetry Generation, Part 1: Hidden Markov Models," there is often no rhyme in the poems generated by our initial HMM model. To introduce rhyme into our poems, we build a dictionary of rhymes that Shakespeare uses by looking at the last words of rhyming line pairs. Then, we train our new HMM model in the reverse direction, and we can generate two lines that rhyme by seeding the end of the line with words that rhyme and doing HMM generation in the reverse direction.

Also, instead of using a poem as a singular sequence as in the initial HMM model, we now use a line as a singular sequence. The first reason is that it seems that the transitions among lines are not well learned even if we use a poem as a singular sequence. The second reason is that it is natural to use a line as a singular sequence if we want to generate two lines that rhyme by seeding the end of the line with words that rhyme.

A sample sonnet generated by the new HMM model using 64 states:

Excuse might with their meadows ignorance,
Figure and is a thus nor this their love,
I what with whence time's bonds therein advance,
And the master of those should that above.
Thee shadows not so when i on thy hooks,
Age with made thy thoughts although soul's to brood,
Hast by robbed but winged upon their looks,
Heart's and in heat fair whilst you hath be blood.
This hold heat look that each wilt rich confounds,
Of true beauty of all from more esteem,
World live him thou with thy despised sounds,
With a quickly pride his antique redeem.
Whose better why sigh the love's sympathized,
Crests the bankrupt i loving thy devised.

Now we have very good rhyme compared to the initial HMM model. This definitely makes our poem more like a Shakespearean sonnet, but there is some tradeoff in quality and creativity as the last words in all lines of the generated poems now must be those words also appear to be the last words of some lines in the training data.

Improving recurrent models:

As we described in Section "Poetry Generation, Part 2: Recurrent Neural Network", the sonnets generated by our initial RNN model consist many non-words. To solve this problem, we replace the one-hot character encoding by embedding word encoding in our improved recurrent model. We implement two different methods to train our embedding word vectors. One method's input and output both are embedding word vector, deleting the softmax layer and using mean_squared_error as our optimization objective. Another method's input is embedding word vector while output is one-hot word, with the softmax layer and using categorical cross-entropy as our optimization objective.

1. Method One (Input–Word Vector, Output–Word Vector)
   RNN structure shown below:

```
# define model
model = Sequential()
model.add(LSTM(90, input_shape=(X.shape[1], X.shape[2])))
model.add(Dense(Dimension))
print(model.summary())
# compile model
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
# fit model
model.fit(X, y, epochs=150, verbose=2)
# save the model to file
model.save('model_advanced.h5')
```

   The corresponding code shown in "LSTM_enhanced_1". A sample sonnet generated by the improved RNN model using Method One:

shall i compare thee to a summer's day,
a she a nothing what heart or or mind,
you me thou behold thine if art and mark me and true departest,
as which eye loves hath a which death departest.
hast who where in heart in nothing departest,
and proud him or upon mind truth eyes departest,
grace can heart sinks her new false day and keep and world beauty need,
by false thee ashes and with departest self.
and which saw that than thee thee though departest and back and keep,
upon o to false and up new since old by and with although new whilst,
shall thee me see thousand he seen heart that and heaven's and some,
eye do will rude from eyes wealth muc.
heart gentle worse fair see hours will you seen,
mine say a night should ever beauty eyes since and bright with nothing.

2. Method Two (Input–Word Vector, Output–One-hot Vector)

```
# define model
model = Sequential()
model.add(LSTM(150, input_shape=(X.shape[1], X.shape[2])))
model.add(Dense(D))
model.add(Activation('softmax'))
print(model.summary())
# compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# fit model
model.fit(X, y, epochs=200, verbose=2)
# save the model to file
model.save('model_advanced_1.h5')
```

The corresponding code shown in "LSTM_enhanced_2". A sample sonnet generated by the improved RNN model using Method Two:

shall i compare thee to a summer's day,
thou art more lovely and thy temperate,
so is shall governs with most my dyer's,
and to might wound a that dear still see.
but hate your time's evil thee no one,
to put looks love love then do sound mine,
suff'ring with show thy time and past my hate,
all should live once in their add their.
to their sweet hast are see doth thine lies,
but you you none so my a greater grief bright,
and to he love with now still dost of not,
that art i hear and thy in time am lie.
thou leap with ranks with his eye,
and one they friend so make a tell decay cold.

Here, we set Temp=1 for the Method Two. Since if we set Temp small enough, then the sonnet generated will be #18 sonnet exactly, which implies strong overfitting.

Sonnets above imply that Method Two is better than Method One. However, we still cannot ensure the right rhymes compared with the improved HMM method.

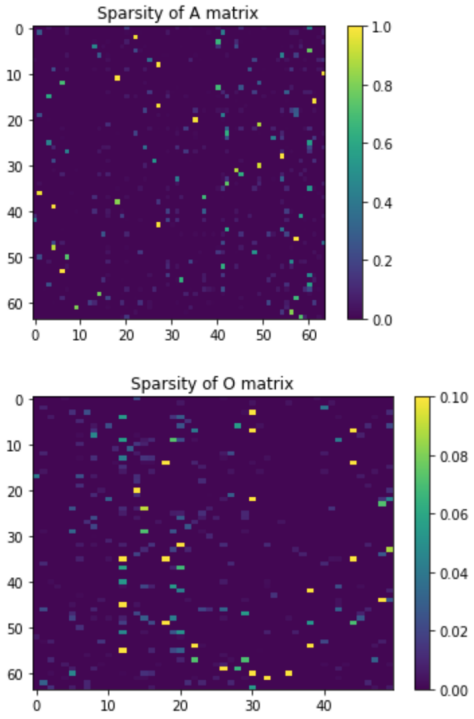
6 Visualization and Interpretation

We focus on the visualization of the poems generated by the rhyme-improved HMM model with 64 states. First, we give a list of the top 10 words that associate with 5 hidden states.

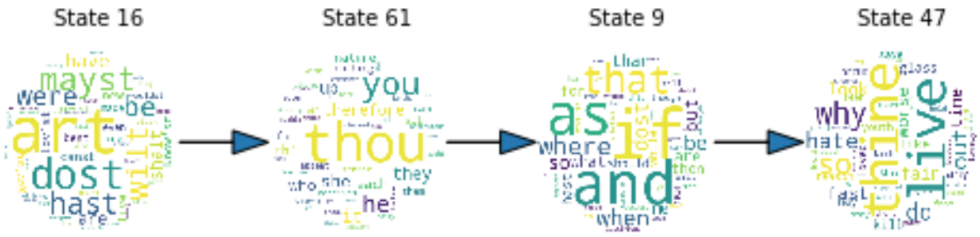
- State 12: part, hand, day, bright, some, state, grow, one, gone, form
- State 15: thee, say, come, heart, good, trust, me, thought, love, thence
- State 16: art, dost, mayst, wilt, hast, be, were, have, are, shalt
- State 36: seen, well, day, new, away, green, abide, twain, grew, past
- State 40: i, and, not, that, this, she, my, what, who, they

We see that some of the hidden states represent parts of speech. Specifically, State 16 represents auxiliary verbs or linking verbs, and State 40 represents pronouns. Some of the hidden states represent number of syllables. Specifically, States 12 and 15 both represent one-syllable word. Finally, State 36 represents rhyme. There are two rhyming pairs (seen, green) and (day, away).

Then, we try to interpret and visualize the learned transitions between states. There are many possible ways to visualizing the transitions. One way is to visualize the sparsity of the trained A and O matrices as we did in Homework 6. The obtained figures are as follows. Note that both matrices are sparse.



Another way to visualizing the transitions is to plot a figure like the animation for "Visualizing the process of an HMM generating an emission" in Homework 6, i.e., plotting all the states using wordclouds and transitions using arrows. However, the resulting figure does not look good since we now have as many as 64 states. Hence, instead, we randomly select an initial state, and then we only plot the transition to the state associated with the largest transition probability, and so on. Thus, we obtain a linear sequence of states, represented by wordclouds as follows.



State 16 represents auxiliary verbs or linking verbs. State 61 represents pronouns. State 9 represents conjunctions. State 47 represents a diverse set of words. Recall that this HMM model is trained in the reverse direction. Thus, it seems that these four states represent "the last word of a line $\rightarrow$ the first word of a line, which is a conjunction $\rightarrow$ a pronoun $\rightarrow$ an auxiliary verb or linking verb," which is grammatically reasonable.