Machine Learning & Data Mining
Caltech CS/CNS/EE 155
Shakespearebot 5000

Joanne Li, Yubo Su

Overview: Glittering Generalities

The objective of the project was to create a program that can generate Shakespearean sonnets at a level better than random guessing. Properties that could be considered include word choice, syllable count, meter and rhyme. In order to do this, a simple unsupervised learning model, the Hidden Markov Model (HMM) was used. By examining various options for preprocessing, we attempted to write a program that would generate a Shakespearean sonnet. Our success hinges on the generosity of the reader.

Preprocessing: Or lack thereof

We preprocessed the data with the primary goal of identifying the same words in the same way (e.g. "Home" and "home!" should be treated as the same word). We reformatted the data in a form that could be easily understood by our parsers (each poem was read in as a single line, with line breaks separated by the '|' character). So that identical words are mostly all identified, we removed all punctuation from the input lines, choosing instead to insert our own punctuation upon poem generation. Toward the same goal, we also turned all words into lower case. Additional preprocessing that would have been useful if the poetry generation could accommodate the information would have been:

- Syllable tagging - Knowing the number of syllables per word would have helped us find line lengths that are guaranteed to be the correct length rather than estimating simply a number of words
- Rhyming class - Each line can be associated with a rhyming class, so that the current line must end with a word in the same rhyming class as the target word.
- Meter tagging - Knowing the emphases on the word syllables would have helped generate iambic pentameter rather than a random collection of words.

Unsupervised Learning: Just like unsupervised kids, doesn't work

We performed our unsupervised learning by applying the expectation-maximization algorithm discussed in class to our HMM until the Frobenius norm of the difference is below some

multiple of the Frobenius norm of the A, O matrices. This multiplier was allowed as a parameter to the learn() function.

It was found that as the number of hidden states was increased, the words associated with emission from each latent state became uninterpretable. Thus, the number of hidden states was kept low, for easier interpretation of shared characteristics. Properties such as syllable count would have required syllable preprocessing to identify, and even more difficult properties such as connotation of words may be impossible to identify with the highly local HMM model.

Visualization: Pretty colors (hopefully)

For visualization, we decided that we would only allow each word to be associated with a single latent state, to handle the problem identified on Piazza where the most common words would be the most common emissions from all latent states. The top 10 words corresponding to 5 (and later 10) hidden states were (note that the '|' character is the end of line character for each poem line):

1. k=5
    - 1.1. ['|', 'of', 'on', 'this', 'thee', 'thine', 'hath', 'world', 'time', 'too']
        - Seems to be objects of sentences with a few exceptions
    - 1.2. ['the', 'with', 'to', 'in', 'his', 'their', 'by', 'make', 'do', 'an']
        - Prepositions generally go here it appears
    - 1.3. ['thou', 'i', 'self', 'love', 'all', 'it', 'be', 'should', 'can', 'what']
        - Subjects of sentences are here
    - 1.4. ['thy', 'my', 'a', 'but', 'not', 'for', 'that', 'so', 'when', 'as']
        - Possessives are here
    - 1.5. ['and', 'or', 'you', 'night,', 'at', 'such', 'eyes,', 'now', 'these', "th'"]
        - Conjunctions are here
2. k=10
    - 2.1. ['thy', 'thou', 'my', 'i', 'your', 'not', 'own', 'be', 'it', "love's"]
        - Possessives
    - 2.2. ['on', 'thee', 'doth', 'time', 'fresh', 'thee.', 'day', 'golden', 'time,', 'decay,']
        - Objects of sentences, with some exceptions?
    - 2.3. ['in', 'of', 'at', 'their', 'should', 'die.', 'beloved', 'use,', 'old,', 'done,']
        - Prepositions
    - 2.4. ['art', 'are', 'shouldst', 'eye,', 'lines', 'well', "beauty's", 'sun', 'muse', 'brow,']
        - Verbs! At least a few. Didn't show up before
    - 2.5. ['love', 'self', 'that', 'sweet', 'than', 'do', 'me', 'times', 'beauty', 'more']
        - Objects
    - 2.6. ['|', 'which', 'treasure', 'these', 'within', 'despite', 'summer', 'gaze', 'please', 'grave']

- A few objects and a few transitions.

2.7. ['thee,', 'night,', 'eyes,', 'prove,', 'would', 'age,', 'me.', 'be,', 'me,', 'thing']

- All end of clauses

2.8. ['and', 'the', 'to', 'with', 'this', 'his', 'by', 'or', 'make', 'thine']

- Not clear

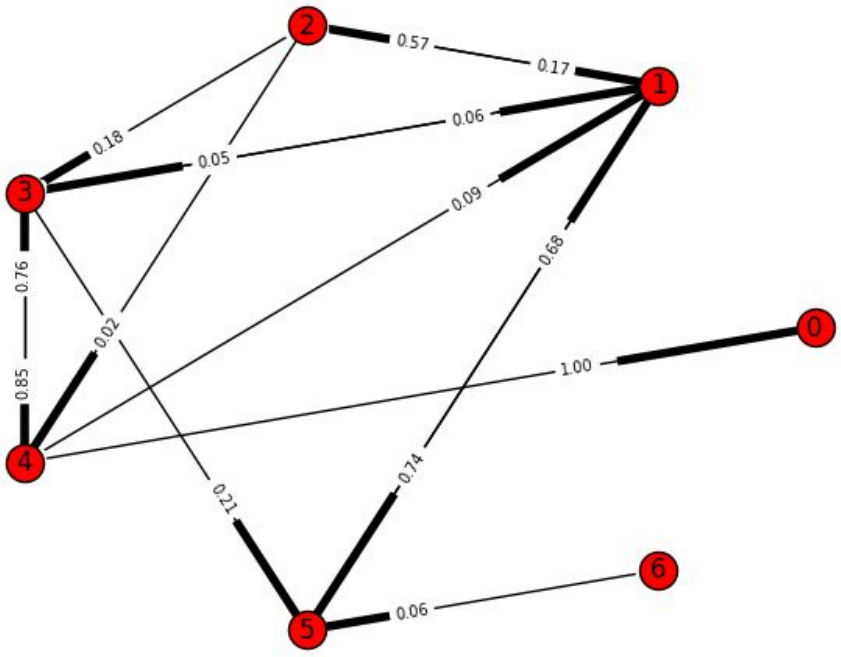
2.9. []

- Empty! Not surprising

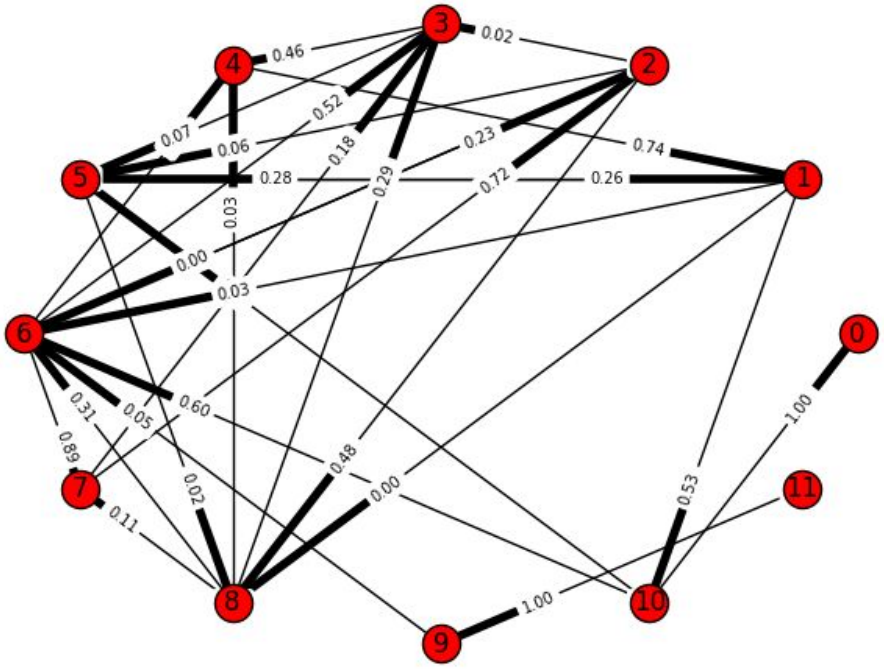
2.10. ['but', 'for', 'a', 'when', 'so', 'as', 'is', 'then', 'if', 'from']

- Conjunctions?

Note that two states are also included (0 and k-1) for the start and end states, for interpretation of the below visualizations.



State transition matrix for 5(+2)-state HMM.



State transition matrix for 10(+2)-state HMM.

A few notable transitions:

- k=5: 3, 4 transition into each other, 1, 5 transition into each other. The interpretation is unclear; perhaps transitions like "thy love" are conceivable for the former. The latter is a bit nicer, since it is conceivable that subjects follow conjunctions, and indeed the 5->1 transition is a bit stronger than the 1->5. We're slightly grasping at straws due to the weak characterization though.
- k=10: 9->11 is a 100% transition, so thankfully 9 has an empty characterization, else every time we transitioned into state 9 we would automatically terminate our poem! Another interesting sequence is 4->3->6 being a strong series of transitions, which corresponds to verb->preposition->[object, transition]. This makes sense for constructions such as "sing for that which…" or "sing for <something>"

Suggested Improvement

We identify a problem with our methodology above: the latent states themselves are not equally probable, thus if a word has unity probability of being emitted by a rare latent state and 0.5 probability of being emitted by a more common one, we should prefer to associate it with the latter hidden state, which will account for the bulk of its emissions. A seemingly reasonable heuristic for the rarity of latent states are the eigenvectors of the A matrix. Since A is a probability matrix, it must have an eigenvector with eigenvalue unity corresponding to the stationary distribution; the components of this eigenvector can be identified with the rarity of the latent states.

Thus, we suggest the following improvement: Instead of identifying each word with the latent state most likely to emit it, identify each word with the product of the probability of observing each latent state with the probability of that latent state emitting the word. In other words, instead of $max_k O_{jk}$ for the j-th word, we want $max_k O_{jk}\pi_k$ for stationary distribution π.

Poetry Generation: Our program wrote some words.

For poetry generation, as described above, our preprocessing was inadequate to capture anything about the meter, syllable count or rhyme of the poem. However, it bears noting that our existing codebase extends easily to doing so should the preprocessing have been performed. Since the Viterbi algorithm is an iterative algorithm in that it takes candidate paths and likelihoods and updates them by appending a new output probability, our code is designed such that each Viterbi update is a single call to a single function. This function also takes a multiplier argument that multiplies the emission probabilities derived from the A, O matrices (computed via memoized forward-backward) before calculating a maximum probability sequence. Thus, within the calling function to the Viterbi update (in our code, the learn() function), we could simply keep track of the meter and rhyme, and should a word need a specific meter/rhyme/syllable count, we can simply mark the words that do not satisfy these requirements with a 0 in the multiplier list and the words that do satisfy with a 1. The conclusion is that our adaptive code design could have facilitated this extension with additional testing and time.

- k=5 poem

  For be canst self shall filled me me,
  Stain see dumb that day who what if,
  But believe scythe all so i no princes,
  Self which give thy thou traffic in these,
  Shall love but all and thy held: own,
  Ten music store yours love sheds ten for,
  Lies than not summer so still let you,
  Threescore paws better me please must by and,
  Flowers deserv'st fore husbandry canopy top therein this,
  Thought sounds predict blood come huge of invocate,
  Time thou till thou more own are every,
  Way sweet good with change thee fair then,
  Thy good as drawn sort in my self,
  Place and sensual compare beauteous changing do morn.

- k=10 poem

  Like own men and her one question that,
  Truly possessed know fond thy back of was,

Question imaginary of though things low light to,
Snowed of i thy the at say to,
Barrenly shame please fortune's did hands and not,
Nature again like time alchemy: thee i feed'st,
Were light make left memory state when poet's,
Orient at not behold pleasure farther great man,
Cruel: black more prove to do cloud faces,
It clouds in hand and or with comment,
To year world of authorizing not when all,
Toiled: sorrow and usurer thy before and in,
Unprovident of from his calls replete for in,
Mistress for thy unthrift must that my are.

These poems were generated probabilistically; rather than running Viterbi for the maximum probability sequence, a selection was made at each step based on the probabilistic weighting that Viterbi predicted. This was chosen because otherwise the poem would strictly emit "thus thy thus thy…" the most common words. We could have also outlawed duplicates, but this would have skewed the poem to use the most common words first, so a probabilistic generation was used.

Development Process: We failed many times

In order to maximize the probability that our code would work on the full dataset, we implemented many trial phases. Some examples of the trial data that we examined were:

- '10101010', '01010101' - The code was able to identify with two hidden states the oscillatory nature of the two emission states.
- '1234567895' - With 5 hidden states, the code was able to identify (n)with (n+4)
- Single/few Shakespeare sonnets - We first tested our code on single and few Shakespeare sonnets before running on the full dataset.

The many iterations of the testing process can be found in our GitHub repository https://github.com/yubo56/CS155SonnetGeneration.

Acknowledgements: Without you we'd be even more hopeless

We thank Prof. Yisong Yue for the wonderful class and the TAs for their fantastic help in explaining and clarifying the material. We also thank William Shakespeare for his generosity in withstanding the atrocities committed in the development and subsequent release of this code.