**EE155: Shakespeare Bot 5000**

*Akshta Athawale, Mannat Singh and Miguel Aroca-Ouellette*

**1. Overview**

For the development of our Shakespeare Bot 5000 our team chose to take an iterative approach, wherein the first goal was functioning preprocessing, HMM training and Naïve Generation upon which incremental improvements were built. The incremental improvements first involved using visualization and qualitative sonnet analysis to determine an appropriate number of hidden states. Following this we added further preprocessing and more complex poem generation which used rhyme, meter and parts of sentence tags to generate a Shakespearian sonnet. Each step was tuned incrementally and qualitative analysis showed that each had an effect in generating a better poem.

We have three members in our team, Akshta Athawale, Mannat Singh and Miguel Aroca-Ouellette. Akshta was in charge of visualization and interpretation. Mannat was in charge of the unsupervised learning and HMM implementation. Miguel was in charge of preprocessing, poem generation and the additional goals. However, there was significant collaboration each task across the team, particularly with regards to debugging and qualitative sonnet analysis.

**Unsupervised Learning**

**2.1 Preprocessing**

Our team chose to tokenize the dataset by word, where a singular sequence would be represented by a line of a poem and each type of stanza (quatrain, volta and couplet) maintains a separate list of tokens. This approach was chosen as Shakespeare uses different intonation and language in each of his stanzas. The quatrain is the body of the poem and introduces the theme or story with lots of adjectives describing the subject of the poem. The quatrains maintain a consistent mood through, except for the volta, which usually represents a change in mood. This is grammatically demarcated by particularly strong adjectives and adverbs, as they have to provide a counterpoint in 4 lines to the 8 of the previous quatrains. By separating the volta from the other quatrains our team hoped to capture this shift in mood, or at least the shift in intensity. Similarly, although the couplet does not bring the same intensity as the volta, it does use different language from the rest of the poem as Shakespeare is bringing a close to the story he has presented. It usually starts with a conjunction (i.e. “or”, “but”, “yet”, “then”) and instead of posing the questions or accusations of the previous lines, it presents a statement or final action. Once again, by separating the couplet from the other stanzas, we hope to capture his literary voice.

Regarding punctuation our team took a simple, but effective approach. When tokenizing the data, all punctuation was stripped except for apostrophes “ ‘ “ and hyphens “-“.This allowed for words implying possession and plural words to stay separate, which should be beneficial as such words appear in different contexts and with different meaning. More importantly, Shakespeare often uses apostrophes within a word as alternate spelling or to skip a syllable, we wanted to keep these words intact and readable when generating our poems; for example, “consum’st” and “murd’rous”.

Hyphens were kept as the hyphenization of two words yields an entirely meaning than the non-hyphenated word, and treating the two words as separate would provide a significantly different meaning, particularly in Shakesperian English; for example, “all-eating” means self consuming, while “all eating” means everyone eats. (Maybe change) Punctuation was then added back in during poem generation.

During preprocessing, we also used the line endings and the known rhyming scheme (*abab cdcd efef gg*) to create a rhyming dictionary which was later used by our poem generation algorithm to ensure rhymes at the end of the lines.

Explain choices

Explain initial choices

Explain what changed

Final choice

Dataset Analysis

**2.2 Unsupervised Learning**

In order to perform unsupervised learning we used the Baum-Welch algorithm as laid out in the lecture slides and HMM notes. The stopping condition was the convergence of the norms of both the state transition and the observation matrix.

How did you choose hidden states

How did you tokenize words

How did you split up data into different sequences

**3. Visualization and Interpretation**

**4. Poetry Generation**

To generate our poems we use three trained HMMs, one for each stanza type, and individually generate each stanza. The stanzas were then combined in the correct order to generate our Shakesperian sonnet. The first letter of each line was capitalized and punctuation at the end of the line was added randomly; with a high probability simply a comma was added, and with a small probability an exclamation mark, question mark or semicolon was added. Using this naïve generation method and 30 hidden states Sonnet 1 was generated below. In the Appendix, Sonnets A1-A5 demonstrate the effect of different number of states on naïve generation.

*Sonnet 1*

*Which upon thinking for their head, (1)*

*But seeing sauces patience do war,*

*And face thy pipe thy if my function swear therefore are temptation invited,*

*But dost and where stain all strange,*

*And that those things wilt thee dreams true, (5)*

*Your very I shall think begins asked than and is needing clock to bright,*

*When lovely reproach harvest defaced;*

*That so admitted lame do altered days,*

*He newer their thoughts of suited point a him proof fear brag that they thee,*

*Against what of gracious the great, (10)*

*And I walks truly by thy lines thine air,*

*That I and doth at of such yet nor to must everywhere,*

*And thy thy you that thy store.*

*For so need besides thou knows mind.*

Although mostly nonsensical, sonnet 1 has a few lines which appear to maintain Shakespeare’s voice and are intelligible, such as lines 1, 5,8 and 11. Overall it also feels like a Shakespearian due to the words used and the overall sentence flow of each line. Particularly it is obvious to note that many of the lines (1,2,3,4,7,10,13,14) start with conjunctions, which one of the grammatical characteristics of Shakespeare’s sonnet writing. The HMM model was able to capture some of the aspects of Shakespeare’s line structure as the hidden states represented a loose grouping of parts of speech tags. This becomes particularly clear in the starting conjunction characteristic. Because this part of speech always occurs at the same place in the sentence the HMM is able to accurately capture and replicate this characteristic, as is clear in our generated sonnet. This relationship is not as clear cut within the sentence where parts of speech, rhyme and meter are much more varied for each word. At the end of sentences HMMs were able to capture the fact that Shakespeare (and sentences in general) rarely ends a line with a preposition or article. This is most likely because states which have a high probability of generating preposition or articles only have a small probability of transitioning to the end state, which would be an expected result of our HMM training.

Unfortunately, it also lacks some of the key characteristics which defines a Shakespearian sonnet. First of all, it does not match the rhyming scheme at all. However, this is not particularly surprising since naïve generation does not take into account rhymes and the probability of randomly generating two end words which rhyme is small. The sonnet also doesn’t have proper syllable counts with lines ranging from 19 syllables (line 3) down to 7 syllables (line 13). Since the naïve generation line stopping condition is simply based on transitioning to the end state and we have assumed the Markov property, this large variety in line lengths is not surprising. Similarly, the sonnet fails to capture the meter of Shakespearian sonnets; line 5 and line 13 are almost entirely composed of stressed syllables.

Comparing this sonnet to Sonnets A1-A5 which were generated with a different number of hidden states we see that the main difference is in the quality of the sentences. Sonnet A5 (50 Hidden States) yielded a similar number of intelligible sentences as using 30 hidden states, while Sonnet A1 has no intelligible sentences. This corresponds with the expectation that a larger number of hidden states better captures sentence structure, but this increase in hidden states is met with diminishing returns and exponentially longer training times. We also note that the increase in hidden states has little or no effect on the quality of the rhyming and meter within the sonnets.

Line 13->7 syllabeles

Line 3-> 19 syllables

Describe poetry generation

Include Naïve sonnet

Comment on quality

* Rhyme, meter and syllable count
* Does poem make sense?
* Maintain voice?
* How does different number of states affect generation (qualitative)

For good qualitie discuss how HMM was able to capture these

**5. Additional Goals**

**5.1 Rhyme**

In order to add rhyming to our Shakespearian sonnets, during preprocessing we generated a rhyming dictionary. In addition, every line of each poem was reversed before being processed by the HMM training algorithm, this allowed us to model reverse transition probabilities (reading sentence right to left) instead of forward transition probabilities (reading sentence left to right).

Using these new state transition and observation matrices we could then generate each line of the poem backwards. This was done by randomly selecting a word with a rhyme in our rhyming dictionary and using that as the seed for the generation of our poem line. If the line, was a rhyming line (i.e. the second *a* rhyme line), then the seed was selected from one of the rhymes of its previous rhyming line (first *a* rhyme line). This was done similarly for each rhyming scheme.

This ended up working very well and yielded much nicer sonnets than the non-rhyming approach. These sonnets also matched much more closely the sonnets written by Shakespeare since the rhyming scheme was an important characteristic of his approach. The only limitation this imposed is that the final word of each line (the seed word in the line generation) had to come from the rhyming dictionary. We felt that although this was a little limiting, since the dictionaries were so large it did not limit the creativity of our Shakespeare Bot too heavily. In the future it would be beneficial to generate rhymes for many of the words in the poems. Perhaps from an external rhyming dictionary source. This would allow for a larger variety in the rhyming schemes and greater variety in each line since there are more possible seed words. See Sonnet 2 below for an example of a sonnet generated once we added rhyming.

*Sonnet 2*

*The are is be from fair seen brains eyes, (1)*

*Though lest shall o'erpressed said,*

*Which where moving spies,*

*My wide a shall the morning hath clock must prime an allayed;*

*To as farther whether o'er refuse streams womb one love, (5)*

*Stealing did to-day the embassage,*

*As since frost and doth above,*

*For my sins which simple doth vassalage,*

*But die pursuit guess day pass and rank back do to knife,*

*Mine shows is be, (10)*

*And when may thy farther most orphans life,*

*Though love for delights deepest his by thee did see,*

*As suborned do art of with old,*

*I this yet doth death be told.*

As intended, sonnet 2 now matches the Shakespearian sonnet rhyming scheme. However, it does not solve the meter and syllable length problem encountered with the naïve generation; line 4 has 14 syllables and line 12 has almost entirely stressed syllables. These issues are addressed below.

**5.2 Meter**

One of issues we noticed with the naïve poem generation was that without control over the meter, the generated sonnets did not match particularly well the linguistic flow of Shakespeare’s sonnets. In addition, often our lines were simply too long and wordy as our line ending condition was based on word count or transition to an end state.

In order to remedy these problems we decided to incorporate syllable counting. Syllable counting allowed us to generate sentences which roughly 10 syllables (as a Shakespearian sonnet should have), by using the syllable count as a new ending condition for our sonnet line generation. Syllable counting was done using the python *NLTK* [1]library, as well as *pyHyphen* [2] for words that are not present in the *NLTK* library. Although the syllable counting is not 100% accurate, it was close enough for our purpose.

Adding syllable counting also allowed us to keep track of the current syllable stress within a poem when parsing the training poem lines. Using this information, as well as syllable stress information from the *NLTK* library, we were able to provide an estimate of the stress for each syllable for each word; combining the syllable counting stress estimate as well as the *NLTK* stress estimate allowed us to show the multiple possible syllable stresses that each word could (i.e. content as in happy, or content as in material).

When generating each line of our sonnets, we ensured that each subsequent word had the correct starting (or ending when generating backwards) syllable stress in order to maintain the expected Shakespearian meter. This was done by pruning all the words which did not have the appropriate ending or starting stress syllable and re-normalizing a copy of the corresponding column of the observation matrix over this new, controlled distribution.

This yielded particularly large advantages in imitating Shakespeare’s style. Although it was slightly limiting in terms of creativity, since we were on average only pruning ~50% of each word from the observation distribution we still had a large subset of words to choose from. This was in some sense actually a good feature as in guaranteed variety of words within a sentence and avoided repeated highly probably words. See Sonnet 3 below for an example of a poem generated using both rhyme and meter.

*Sonnet 3*

*Which brief were kingdom my thy I is smell! [10] <0,1,0,1,0,1,1,1,0,1>*

*Bad picture's thousand usest and your thy deem, [11] <0,1,0,1,0,1,0,0,1,0,1>*

*If praise the image for my ever tell, [10] <0,1,0,1,0,0,1,1,0,1>*

*Persuade in are muse proud-pied the seem, [10] <0,1,0,1,0,1,0,1,0,1>*

*Them purple green my once forget in keep; [10] <0,1,0,1,0,1,0,1,0,1>*

*Revenge alack but with feathers to your eye, [11] <0,1,0,1,0,1,0,1,0,0,1>*

*Eat harder how if poor to well to weep. [10] <0,1,0,1,0,1,0,1,0,1>*

*Ay laughed still thinking not would with can fly, [10] <0,1,0,1,0,1,0,1,0,1>*

*New but to of addition mistaking, [10] <1,0,1,0,1,1,0,1,0,1>*

*My graciously is health familiar be, [10] <0,1,0,1,0,1,0,1,0,1>*

*Thy graciously against had thy making, [10] <0,1,0,1,0,1,0,1,0,1>*

*Nor horse cool bring for reason you poor see, [10] <0,1,0,1,0,1,0,1,0,1>*

*Skill by ne'er leads o put then than and end, [10] <1,0,1,0,1,1,0,1,0,1>*

*High spent and swear but for what dear a friend. [10] <0,1,0,1,0,1,0,1,0,1>*

Using meter and syllable count consistently generated better poems, as shown by Sonnet 3. The new restrictions do not appear to have affected the intelligibility of the poem and have aided in generating a more accurate Shakespearian sonnet. We’ve manually annotated the number of syllables by each line as [-] and the stress of each syllable as <-> where 0 represents unstressed and 1 represents stressed. As we can see the syllable count is now almost exact to the characteristic 10 syllables. The few differences arise due to words which are not in the *NLTK* syllable dataset and are not properly parsed by *pyHyphen*, for example usest in line 2 is two syllables, while our automated algorithm labelled it as one. Similarly, the stressed/unstressed syllable pattern is thrown off by words that are not in the dictionary or were not properly labelled by our automated algorithm. Unfortunately, the mislabelling of syllable stress can cause a cascade effect as each word is generated based on the previous words stress. A good example of this in the above poem is the word “o” in line 13, which is mislabelled as being high stress while it should be low stress. This mislabelling cascades and the next 4 words are generated with the wrong stress pattern (recall that line generation is done from right to left). However, overall taking into account the meter of the poem was very beneficial to replicating the Shakespearian sonnet style.

**5.3 Part Of Speech (POS) Restraints**

We noticed that one of the grammatical difficulties which we were sometimes running into with the sonnet was repeated part of speech (POS) tags. For example, in the generated sentence “Level *inspire* *convert* device perish”, both “inspire” and “convert” are verbs and it is grammatically quite rare for this to happen; one exception would be the sentence “He *began* *working* on his project”. As such, as a general rule we decided to enforce that POS tags of one type could not be repeated sequentially; i.e. a verb could not follow a verb. We also enforced that certain tags could *only* occur sequentially (i.e. an adverb must lead a verb and a pronoun must lead a noun) and that certain tags could never occur sequentially (i.e. a preposition should never lead a verb). POS tagging was done using the *NLTK* library and the data was then stored in a dictionary for quick retrieval. Many of the tags provided by the *NLTK* library were too descriptive for our purpose, so they were grouped into broader tags; i.e. “Determiner”, “Predeterminer” and “Wh-determiner” where all grouped under “Determiner”. See Sonnet 4 below for an example of a sonnet generated using rhyming, meter and POS retraints.

*Sonnet 4*

*Death o and second mountain may no eye;*

*Upon is dignifies only anew,*

*Cost and dost do brand and to history,*

*Help kiss deserv'st thinks to keep thy hue,*

*Cheered worth's contend then did which fresh self is,*

*But which of well-contented endured,*

*Fresh pity more wink root to heinous this;*

*Breathed such as put and time were much assured,*

*In dumb the others sweet and lives wouldst bright,*

*With gazers thee but triumphant with expressed,*

*She health growing and words in eyes of night,*

*Them prouder sweetly gaze or out unrest!*

*Which untrue silent fool is dear love thee,*

*Love beauty after an by be the me.*

It’s much harder to evaluate the effect of the POS tagging restrictions, however it appears that in general this avoids some common grammatical mistakes which previous versions of our generation algorithm were encountering. For example, in Sonnet 2 line 10 we have the double verb “is be” or in line 13 we have the double preposition “of with”. Overall, this approach yielded favorable results and qualitatively generated grammatically correct sentences slightly often than before. In addition, it had the added benefit of adding more variety within the poems as highly probably word sequences were now broken by the POS tag of the current line’s randomly generated seed word. Since there are so many words for each POS tag this did not noticeably affect the creativity of our poem generation algorithm.

**5.4 Additional Texts**

One of the things we learnt from the Kaggle project was that having a larger dataset is often beneficial. There was not a specific problem we were trying to solve with this addition to our Shakespeare Bot, but we felt that perhaps the larger dataset and rhyming dictionary would be beneficial in adding creativity to our poetry generation and counteract some of the limitations imposed by rhyming and meter.

We added the Edmund Spenser dataset as well as sonnets we fetched ourselves from some of Shakespeare’s contemporaries which abided by the Shakespearian sonnet structure; these included John Keats, John Clare and Robert Frost. The result of our expanded sonnet dataset can be found below.

* What were you trying to fix
* How did you attempt to fix it
* Why did you think what you tried would work?
* Did it succeed?
* If not why?
* What tradeoffs in quality and creativity did this change cause

**6. Conclusion**

In conclusion, our team was successful in implementing a Shakespeare Bot which produced Shakespearian sonnets of a reasonable quality. We took several steps to improve upon the basic HMM implementation and naïve poem generation, and each modification had a visibly positive effect on the quality of the poems. Rhyme scheme and meter were the most important in replicating the Shakespearian style, however controlling for POS was the most important in improving the reliability of generating grammatically correct sentences.

**7. References**

|  |  |
| --- | --- |
| [1] | NLTK Project, "Natural Language Toolkit," 03 03 2016. [Online]. Available: http://www.nltk.org/. [Accessed 10 03 2016]. |
| [2] | D. Leo, "PyHyphen 2.0.5," 2014. [Online]. Available: https://pypi.python.org/pypi/PyHyphen. [Accessed 10 03 2016]. |

**8. Appendix**

*Sonnet A1 (5 Hidden States)*

*From else alone know,*

*Time's yield my call of to esteemed wantonness thou without my jewels you more,*

*This first crawls gentle with are stick'st wisdom excellence releasing,*

*Worth hate tables why thy so a speed thy know minds,*

*Art betwixt in thinly in be,*

*And in is self decrees will the thee the me and form self some so,*

*That betraying return fear with forgoing ear.*

*When it extreme turns me within trust sound me,*

*From proof a fair?*

*O those it praise thou all thither tie thine of love.*

*Cloud something as if new sing lovers rhetoric kind,*

*That every my new self in lost sweet doth but kind life rank bright,*

*No things so thy you takes rich my told?*

*Thy this from own by women's king have to to begin.*

*Sonnet A2 (15 Hidden States)*

*How that mine define,*

*Growth guilty the stop golden love I bring consum'st each bear heaven I is he a old fair time;*

*While fitted,*

*But in hand end,*

*That sweet late away cries,*

*Or fear on man although for be function his increase,*

*Numbers so I laid helen's with not grant motion fortune sweet-seasoned yet fearfully with do not,*

*And doth too are of few me beauty his thing white use?*

*To a to my just war works fair pay but the eye thy aye look drawn doth she in lawful it first self fair cured,*

*And time's judgement despise,*

*From lest kind my on say die never like wardrobe it thy,*

*Shall all-oblivious my so!*

*The thou want truth matter,*

*Where not winter thence virtue assure with thus as tongues heats moan.*

*Sonnet A3 (20 Hidden States)*

*Will therefore I the ornament as for of breast fading own excel,*

*Th' didst ripe way,*

*The form a be store,*

*That merit wires eye riches plead thee deeds,*

*For the hand of much be thee!*

*And purpose bide,*

*And thy these a have but that sight,*

*Ah those beauty to to live,*

*No wretch with cured,*

*Though black reckon obsequious all did shall well your with wrinkles beauty my all every me well of fits;*

*But locked on thence daily why hand wondrous quest youth,*

*But where that the rarities ornament,*

*Thrice in thou slander yet my I be so,*

*To him sleep is a men doth right the am or and this wealth I hearsay longer.*

*Sonnet A4 (25 Hidden States)*

*In hath abundance mistress make woman by december's once counterfeit,*

*My used be captain two would by deserv'st to treason,*

*Yet nor be frame of think view,*

*Fairing with sad lace leave confounds deeds are my holds should be words?*

*Look sweetest wood's of single forgotten!*

*Since services write are;*

*Thing of tie me masked and esteem,*

*If a which the wrinkles above,*

*So my smell seeing either return I will graces more,*

*But canst who to bud ears will own,*

*The thought of all taught stol'n world care,*

*Sets all now building more,*

*I thine be me,*

*When you outright with blot.*

*Sonnet A5 (50 Hidden States)*

*Which towers me hath dwell,*

*Who lusty predict think that dull war travel that my i'll black like maturity pilgrimage,*

*What's oft why a when great altered springs to rider wand'ring impart,*

*Profitless the basest are that the those thou self-substantial dulness,*

*Whose which olives from them with prayers ill,*

*The oft thy are oaths prizing misplaced,*

*Music have to not all eternity away,*

*Distilled heavy diseased night which some preserve gentle flower,*

*But do false words new,*

*Shall they thee rhyme,*

*But thy must of me are tell?*

*Yet what foul your of grow grief,*

*That eye mine thy ne'er thee beauty love love of thee,*

*'tis thy must friend go belong.*