Write-up: Text Mining Analysis of Shakespeare Sonnets

Project Overview

I analyzed a collection of Shakespeare's sonnets. First, I analyzed them for word frequency. Then I did a Markov Chain analysis that would generate new sonnets. My first idea was to combine epics like Beowulf and the Aenid, but the language was so different between the two and the spelling so strange that it was impossible to switch between the two at all. Many sonnets (especially love sonnets) are close enough in composition to Markov Chain, so that was my goal. I also generated some sonnets that were a combination of Petrarch's and Shakespeare's sonnets.

Implementation

Making a dictionary of prefixes and suffixes. I started with a general word frequency dictionary, which processed every line and every word in the line. However, looking at prefixes, I couldn't only use one line, because the prefix to the first word in the line would be in the previous line. Therefore I created a variable line_hangover that took the last *n* words of the line and add them to the next line to analyze for prefixes. For processing prefixes, I used a dictionary where both the keys and values were tuples. I struggled a bit before coming to this conclusion, but I realized that tuples were the best answer. They are immutable, so they can be used as keys for the dictionary. On the values side, tuples were useful because they can be easily converted into a list to take a random item from.

Building the sonnet. Once I had the dictionary of prefixes and suffixes, building the Markov Chain was relatively easy. You start with a random key that keeps randomizing until it becomes a reasonable beginning of a sentence (ie the first letter is capital) with the use of a while loop. The program then looks for that key's tuple of values and chooses a random suffix from there. The program continues to do that until it has a sonnet. As far as building the format of a sonnet goes, I kept track of the number of words in a line and the number of lines that have been printed. Once the length of the line was greater than six words, it will start a new line once it encounters punctuation or reaches the maximum of 9 words. It would be difficult to adhere to strict 10-syllable lines because it would have to analyze the syllables, but counting words get it reasonably close.

Implementation

Several things are interesting about creating a randomized sonnet. All sonnets are in iambic pentameter (10 syllables, in a pattern of unstressed-stressed), and I was expecting that it would be tough for the Markov Chain to accommodate that. It turns out, however, that since the chain can take more than one prefix, even using 2 words as a prefix creates sonnets that makes the result incredibly close to this meter. Even though for the most part, the sonnets are nonsense, they retain the mood of the original pieces. The following are some examples that were generated. Black words are transitions, colored are from one specific sonnet.

Shakespeare's sonnets only, prefix length one.

Praising thy Will, And do not free.
Whoever hath thought is black, And yet I send
Them where buried where late the even,
Doth homage to prove thee all ornament,
Itself so that liberty commits, When every eye is

Thine eye, As those friends hid in themselves their
Masked buds doth use And simple truth would willingly
Impart: O! but as the meadows green,
Robbing no blot? Thou art as all these,
For my heart, forbear to know sweet up envy,
Evermore acknowledge thee, the lines and this growing age,
Nor draw no cure, the child of better equipage:
But best acquainted, Upon the rest defeat,
To speak ill report. O! in this,
And therein showst Thy looks with loss.

Shakespeare's sonnets only, prefix length two.

For truth proves thievish for a look;
Possessing or pursuing no delight, Save what is told.
Thy glass will show thee me. If the true
Concord of well tuned sounds, By unions married,
Do offend thine ear, They do but sweetly chide
Thee, who confounds In singleness the parts that thou
Shouldst not abhor my state: If thy unworthiness raisd
Love in love's fresh case, Weighs not the disgrace:
Nor can I grieve at grievances foregone,
And heavily from woe to woe tell oer The
Sad account of hours to crave, Being your vassal,
Bound to stay your leisure! O! let me,
True in love, to thee which thou receivst not
Gladly, Or else receivst with pleasure thine annoy?
If the dull substance of my breast.

Shakespeare's sonnets only, prefix length three.

I am shamed by that which I compile,
Whose influence is thine, and born of thee:
In others works thou dost but mend the style,
And arts with thy sweet graces graced be;
But thou art all my art, and dost advance
As high as learning, my rude ignorance.
Whilst I alone did call upon thy aid,
My verse alone had all thy gentle grace;
But now my gracious numbers are decayd,
And my sick Muse doth give an other place.
I grant, sweet love, thy lovely argument Deserves the
Travail of a worthier pen; Yet what of thee
Thy poet doth invent He robs thee of,
And pays it thee again. He lends thee virtue,
And he stole that word From thy behaviour.

As you can see, a prefix length of two produces the best results that are somewhat intelligible but not solely from a couple sonnets.

Here's a sonnet made from *Hamlet*, mostly nonsense.

Hamlet. O that this player here, But in battalions!
First, her father slain: Next, your son gone;
And he most violent author Of his own petard:
And t shall go hard But I am satisfied
In nature, As by lot, God wot,
It came to see your kingly eyes:
When I said Man delights not me?
To think, my lord, you know right well you
Did; And with a general groan. Arm you,
I pray God. God b wi ye!
O, what a treasure hadst thou! What treasure had
He, my lord? Why, as a stranger give it
Smoothness. O, it offends me to the Dane.
O, farewell, honest soldier; Who hath relievd you?
Bernardo has my place. What, is Horatio there?

Here's one from Petrarch's sonnets, with prefix length two.

She caught, then bound him fast; then such revenge
She took the right path if e'er I went
Astray, It grieves me so, My only sun my
Fond but harassd heart! Yestreen we left her native
Spheres, To trace a loveliness this world be won,
Living on wishes hopeless to acquire. Yet,
Were the spoils Of envious time, and to Thules
Coast, Nile, Tanaïs, and Calpe should resound,
And dread Olympus. But a foul dungeon to the
Spot whence came the genial day That gave me
The slave Of the soft grass, and violets purple
Light, Move, as an angel upon radiant wings,
New lighted on that fair and strong,
And great, He, in her home is there,
Nor can the tongue, or what destiny Will give.

And one combined from Petrarch and Shakespeare, with prefix length two.

Ere medicine old or new, Nor ever wind rustled
So verdant leaves, As were by me be borne
Alone. In our two loves there be Who in
Our faults by lies we flatterd be.
O! call not me to that fair breast combined,
Beauty and Virtue yet they dwelt so bright,
That neer rebellion ruffled that pure spirit,
Pitying my woe, Had not appeard to me.
That thou art assured mine; And life no longer
Meet in rhyme, Pray that I prove,
To her are trusted by one sovereign power.
Hence tis, wheneer my lips would silence break,
Scarce can I more, Than this rich praise,
That you alone, are you? In whose confine immured

Is the crown and palm of good or evil.

Reflection

It worked decently well to work off the word frequency template. Although that did lead to unnecessary computation running the code, it barely contributed to run time. There are numerous reflections I could make to the code. One would be to improve the accidental capitalization in the middle of lines. I would also try to construct a better way to determine line length, possibly by using syllables. I did not have a good plan for unit testing. I tested as I went with a really really short poem of my own creation to test word frequency and Markov Chaining, but I did not write them into docstrings. After randomizing some things, it is tough to do unit testing at all. I think taking a long look at Beowulf and the Aenid would have helped determine earlier that the combination of the two wouldn't really work out. Going forward, I'm proud of how I figured out how to use tuples and dictionaries effectively. I learned how to index an element of a tuple inside a tuple, for one.