Michael A. Schulze

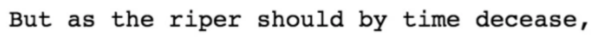
Iambic Algorithm

(https://michaelamadorschulze.files.wordpress.com/2016/07/shakespeare_glasses.jpg)

I first fell in love with poetry, particularly sonnets, while I was in my high school english class. I remember loving them so adamantly because they felt like building a puzzle backwards, similar to the beauty of a streamlined mathematical equation with all it's little intricacies. I recall when I explained this enthusiasm to my professor she said, "It's great you think about sonnets like that, but once you delve deeper into poetry it becomes too complicated to think about it in a mathematical way." As a budding data scientist I thought I would test this theory of hers and find out if a sonnet can be broken down into a mathematical equation.

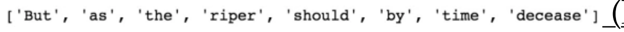
A Shakespearian sonnet is made up of fourteen lines written in Iambic pentameter and a rhyming scheme of abab, cdcd, efef, gg. Iambic pentameter is a poetic system that focuses on the stress of words. Here is a basic example "da dum da dum da dum da dum da dum. " As you might have noticed when reading the line 'dum' has more stress than 'da' causing a rhythmic pattern, iambic, in the sentence. Pentameter means five 'feet', in this case each foot is 'da dum' leaving us with 5 pairs of two syllables one stressed and one unstressed in each line. This is the most basic form of a sonnet, but without going into too great detail there many exceptions such as feminine endings, inversions, caesuras, falling meter, and each word can have a different stress.

There is a dictionary created by Carnegie Melon University that has all the different pronunciations of many english words. Within that information, they have the stresses within each word. To use the CMU dictionary first, I needed to break up each individual word and remove all the unnecessary punctuation. In this case, here is a line from Shakespeare's sonnet number 1.



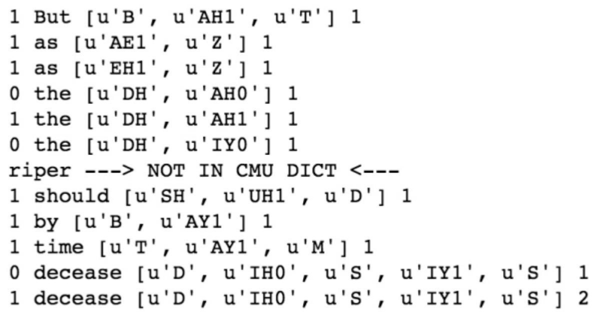
```
But as the riper should by time decease,
```

(https://michaelamadorschulze.files.wordpress.com/2016/07/sentence.jpg)



```
['But', 'as', 'the', 'riper', 'should', 'by', 'time', 'decease']
```

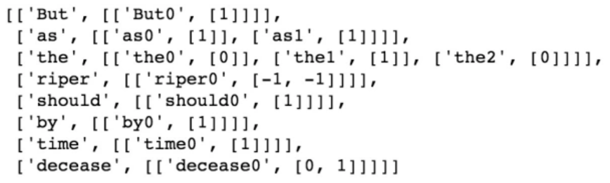
(https://michaelamadorschulze.files.wordpress.com/2016/07/words.jpg)



```
1 But [u'B', u'AH1', u'T'] 1
1 as [u'AE1', u'Z'] 1
1 as [u'EH1', u'Z'] 1
0 the [u'DH', u'AH0'] 1
1 the [u'DH', u'AH1'] 1
0 the [u'DH', u'IY0'] 1
riper ---> NOT IN CMU DICT <---
1 should [u'SH', u'UH1', u'D'] 1
1 by [u'B', u'AY1'] 1
1 time [u'T', u'AY1', u'M'] 1
0 decease [u'D', u'IH0', u'S', u'IY1', u'S'] 1
1 decease [u'D', u'IH0', u'S', u'IY1', u'S'] 2
```

(https://michaelamadorschulze.files.wordpress.com/2016/07/cmu-dict.jpg)

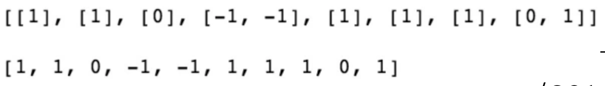
The dictionary is not entirely complete and it doesn't recognize colloquial or uncommon words. As a result I created a function that counted the number of syllables in each word so that I could assign a null value to each of their stresses. Unfortunately, I quickly found my function missed a lot of exceptions in the english language, but it was able to estimate colloquial words quite well. After searching, I found a function written by M. Emre Aydın that counted the syllables of those exceptions, enough to meet my needs. So I merged our two functions together to make an overall more accurate function. Here you are seeing how I collected each unique word and then each different version of that word with different stresses.



```
[['But', [['But0', [1]]]],
 ['as', [['as0', [1]], ['as1', [1]]]],
 ['the', [['the0', [0]], ['the1', [1]], ['the2', [0]]]],
 ['riper', [['riper0', [-1, -1]]]],
 ['should', [['should0', [1]]]],
 ['by', [['by0', [1]]]],
 ['time', [['time0', [1]]]],
 ['decease', [['decease0', [0, 1]]]]]
```

(https://michaelamadorschulze.files.wordpress.com/2016/07/refine.jpg)

Now, I needed to find which version of each word worked best to form a set of optimal iambic pentameter. To do this I made a rating function that tested each version of the word to find the one that fit the meter the best. Here is the model's best guess at the line's meter.



```
[[1], [1], [0], [-1, -1], [1], [1], [1], [0, 1]]
[1, 1, 0, -1, -1, 1, 1, 1, 0, 1]
```

(https://michaelamadorschulze.files.wordpress.com/2016/07/final2.jpg)

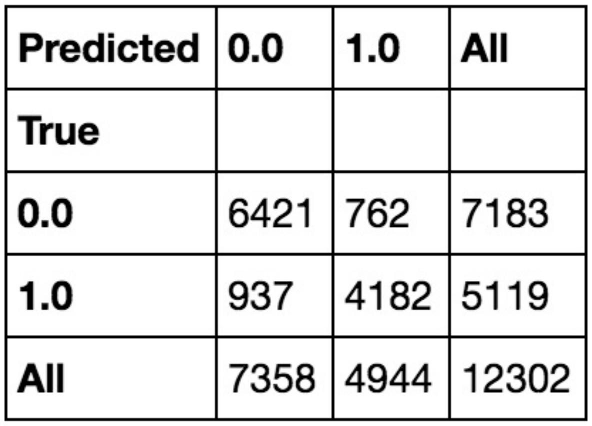
Some of you might notice this is not entirely correct, the words 'But' and 'by' are commonly read as unstressed. This is caused primarily by the uncompleted dictionary, as seen earlier both words are missing a version where they are unstressed, but this can also be due to the fact that some words sound unstressed when next to words which are more stressed. Unfortunately, without a more complete CMU dictionary it is impossible to get the meter perfectly. That being said, as long as I am close enough, I should still be able to accurately predict which line is text or in iambic.

Most of the breaks in the 'rules' for iambic pentameter are done by the poet for emotional emphasis. As a result, I wanted to try to include a little sentiment analysis of each line to see if that has an impact in predicting the iambic meter. I used a tool called textblob, which gave me the sentiment, from -1 to 1, and subjectivity from 0 to 1. In the future. If I wanted to more accurately represent this variable I could do more in depth sentiment analysis to see which groups of emotions cause these breaks in the 'rules' more often.

To populate this dataset I decided to use a number of different poets including Shakespeare, Keat, Frost, Shelly, Jackson, and a few others; then for non-iambic text I used Jane Austen's 'Pride and Prejudice' and Doyle's 'Sherlock Holmes'. With all my data now cleanly tucked away into my dataframe all I had left was to do a little statistics.

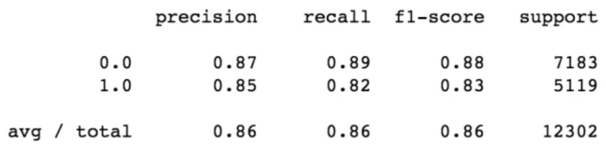
I started with a simple logistic regression with my dependent variable being if it is or isn't a sonnet and for my independent variables I used the number of syllables, the polarity of the line, the subjectivity of the line, and of course the meter for each syllable. Using that formula I got a cross validation score of 0.85978 with a baseline of 0.41611. I suspected I could improve this score by adding interactions between two stresses since each metric line has a pattern. So by using these interactions between the stresses in my logistic regression I was able to improve my cross validation score to 0.87766.

I checked my confusion matrix to make sure I was predicting similar results for both negatives and positives. For positives I was correctly predicting 0.81695 and for negatives 0.89391.



Predicted	0.0	1.0	All
True			
0.0	6421	762	7183
1.0	937	4182	5119
All	7358	4944	12302

(https://michaelamadorschulze.files.wordpress.com/2016/07/cofusion.jpg)



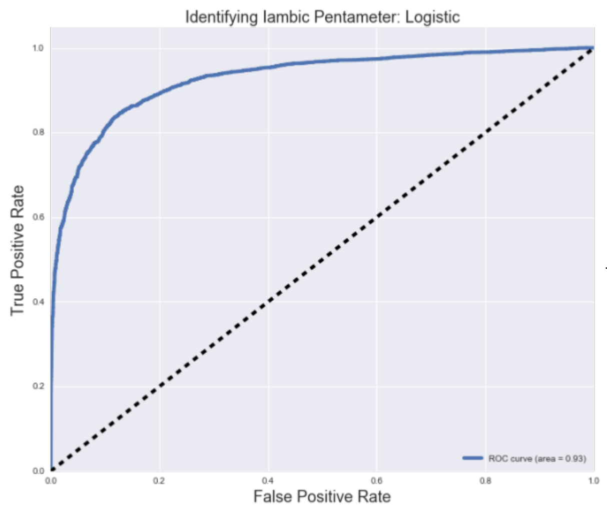
```
             precision    recall  f1-score   support

        0.0       0.87      0.89      0.88      7183
        1.0       0.85      0.82      0.83      5119

avg / total       0.86      0.86      0.86     12302
```

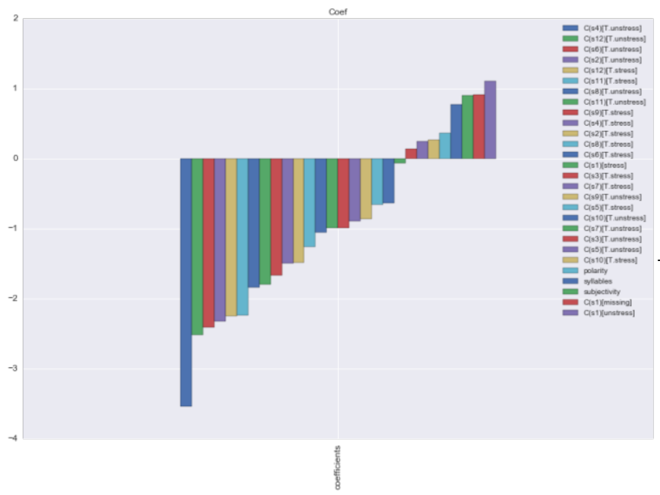
(https://michaelamadorschulze.files.wordpress.com/2016/07/precision.jpg)

After a positive result with the logistic regression, I wanted to see if other more complicated models would have improved results. After using a decision tree and a Naive Bayes classifier and getting similar or poorer results, I decided to settle with logistic regression. Below is the ROC curve for my model with a right angle in the upper left corner being perfect predictions and the dotted line being 50/50 guesses.



(https://michaelamadorschulze.files.wordpress.com/2016/07/roc.png)

I preferred logistic regression because unlike many unsupervised learning models I can see exactly how much each variable effects my result. As you can see, there are stronger variables depicting what is not iambic than what is, which is why I was able to better predict negatives versus positives. The four strongest variables with negative coefficients were all unstressed even syllables. Normally the even syllables are stressed so this made sense to me. Although, I was surprised an unstressed fourth syllable had much more weight than other variables. Of the four strongest variables with positive coefficients, one was from sentiment analysis, number of syllables, and the other two were null and unstressed first syllables. For a normal sonnet the first syllable is unstressed, but with a variant called falling meter the first syllable is specifically stressed which is a fault in my model. Most sonnets have 10 syllables, but a few have 9 or 11 so if it got a line with 7 or 14 syllables it would make sense why syllables would be a good variable. From my sentiment analysis, I used the amount of subjective language in the line of text. This seemed to be a good indicator because, I suspect, poems in general tend to be more subjective versus normal day text.



(https://michaelamadorschulze.files.wordpress.com/2016/07/coef.png)

Overall, I was really happy with my model. I know it can be improved by checking if any lines classified as text accidentally have iambic pentameter in them. Next I will increase my sample size and try using text from twitter, news articles, and speeches to see if my results vary. Lastly, I would test if different authors, journalists, poets, and speakers use meter differently and how that effects the popularity of those works.

Here are a few links if you want to learn more…

https://docs.google.com/presentation/d/1eZj5AxfYo6afV9IWd1V13-xAX1lF-3_OaUDpN9NWlc4/edit?usp=sharing (https://docs.google.com/presentation/d/1eZj5AxfYo6afV9IWd1V13-xAX1lF-3_OaUDpN9NWlc4/edit?usp=sharing)

https://github.com/amadorschulze92/Iambic_regressor (https://github.com/amadorschulze92/Iambic_regressor)

Blog at WordPress.com.