Metrical Analysis of Poetry Using a Bidirectional LSTM Recurrent Neural Network

Brandon Ustaris
Department of Computer Science
University of California, San Diego
La Jolla, CA 92093
bustaris@ucsd.edu

Brian Preskitt
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093
bpreskit@ucsd.edu

Brian Wilcox
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093
bpwilcox@ucsd.edu

Derek Tran
Department of Computer Science
University of California, San Diego
La Jolla, CA 92093
dtt018@ucsd.edu

John Clara
Department of Computer Science
University of California, San Diego
La Jolla, CA 92093
jclara@ucsd.edu

Abstract

In this paper, we propose using a biLSTM neural network to perform supervised learning on English metrical poetry. First we walk through a background on metrical analysis. Then we explain the current metrical tools and research in scansion using machine learning. Finally we describe the tools and data we use in order to learn how to scan English metrical poetry. We will rely heavily on Agirrezabal1 et al's research in "Machine Learning for Metrical Analysis of English Poetry" 2016.

1 Introduction

In the present day, there have been many tools created to automate reading of text out loud. However one can normally tell the difference between when a real human reads a passage of text versus a synthetic voice. After doing some tests as to why a synthetic voice sounded so unnatural compared to a real human's voice, we found a variety of reasons such as the timbre, speed, tone, and stress of the voice when saying words. For the time given to us for this project, we could only focus on one of these aspects and chose to see how well a neural network could learn the stress of syllables of words. If a neural network is capable of learning the stress of syllables, then we are one step closer to creating a synthetic voice that can match a human's voice.

We decided to use English poetry as our dataset due to the many complex combinations of stressed and unstressed syllables used to create rhythm in every line. Learning the stress patterns of poems is a lot harder than learning the stress of normal spoken English, so a neural network trained to do the former would definitely be capable of doing the latter.

Scansion is the process of parsing out the stressed and unstressed syllables in each line. Most English poetry starts with a base pattern or meter on which the poet then chooses to make variations. Consider the first line from Shakespeare's Sonnet 18:

/	/	x	/	x	x	x	/	x	/
Shall	I	com	pare	thee	to	a	sum	mer's	day

Here '/' signifies a stressed syllable and 'x' signifies an unstressed syllable. All of Shakespeare's Sonnets follow the iambic pentameter meter. The iambic pentameter line is comprised of 5 couplets of unstressed and stressed syllables: x / x / x / x / x /. However, the first line of Sonnet 18 deviates from this meter by using a stressed 'Shall' and unstressed 'to'. Due to these variations, metrical analysis of English poetry is a non-trivial task.
There has already been research conducted on this task by Manex Agirrezabal, Inaki Alegria, and Mans Hulden in their paper "Machine Learning for Metrical Analysis of English Poetry". However the paper focused mainly on machine learning tasks and only mentioned a recurrent neural network as an improvement that they would have liked to try. They also performed a lot of preprocessing of the input data that we hope to be able to avoid through the construction of our neural network.

2 Background

A number of tools exist for metrical analysis based off of linguistic knowledge of scansion rules. *ZeuScansion* (Agirrezaball et al 2016) is the latest tool in this vein for English poetry.
Others have started to use machine learning to automatically parse metrical poetry into stressed and unstressed syllables. First, Estes and Hench 2016 used a CRF model to learn Middle High German epic poetry. MHG meters not only use a combination of stressed and unstressed syllables but also uses long and short syllables. They used 750 lines over 3 different texts as their training data as well as 75 lines from a separate text for their test data. Then they performed syllabification on the data as well as selecting features they thought were informative for scansion. They achieved an F score of .904 on their hold out set.
Inspired by Estes and Hench's good results, Agirrezaball et al 2016 tested a number of different machine learning techniques for English language poetry including Support Vector Machines, Perceptrons, Hidden Markov Models, and Conditional Random Fields. Again, they extracted features that they thought were informative for scansion. They achieved results significantly better than the *ZeuScansion* tool.
Graves and Schmidhuber 2005 used LSTMs and bidirectional LSTMs for phoneme classification. They did not have to do feature extraction and found that while bidirectional LSTMs performed better, unidirectional LSTMs also performed well on the phoneme classification problem. We believe that the phoneme classification problem is a harder version of the metrical analysis problem. If an LSTM can perform well on phoneme classification, it should also be able to perform well on metrical analysis.

3 Dataset

The dataset that we used came from: For Better For Verse which can be found from this link `https://github.com/waynegraham/for_better_for_verse`. For Better For Verse is a project by the University of Virginia's Department of English and provides the data in the form of English language poems together with their metrical scansion. Before we were able to train our network on the dataset, we needed to make some formatting modifications to the data in order for it to match up with how we wanted to read in the data. The original data only had the stressed and unstressed marks for each syllable. Since we were attempting to read in the data character by character without syllabication, we needed to add in a neutral mark (using the character ".") to go with every other character. In total, we had around 80 poems in our data set and we split it to have 80% (64 poems) in the training set and 20% (16 poems) in the validation set.

Below is a sample of the original data from For Better For Verse:

```
Slow, slow, fresh fount, keep time with my salt tears;
++++++--++
Yet slower, yet, O faintly, gentle springs!
```

```
-+-+++-+-+
List to the heavy part the music bears,
+--+-+-+-+
Woe weeps out her division when she sings.
+++--+-+-+|+++--+---+
Droop herbs and flowers;
++-+-
Fall grief in showers;
++-+-
Our beauties are not ours. O, I could still,
--+--++-+--+
Like melting snow upon some craggy hill,
-+-+-+-+-+|-+-+--++-+
Drop, drop, drop, drop,
++++
Since nature's pride is now a withered daffodil.
-+-+-+-+-+-+
```

Below is a sample of the modified data to work with our network.

```
Slow, slow, fresh fount, keep time with my salt tears;
..+.....+.....+....+......+....+....-....-..+....+....
Yet slower, yet, O faintly, gentle springs!
.-....+.-....+...+..+....-...+...-....+....
List to the heavy part the music bears,
.+....-...-..+..-..+.....-..+.-...+....
Woe weeps out her division when she sings.
.+...+....+....-...-.+.-.....+....-..+....
Droop herbs and flowers;
..+....+....-.....+.-...
Fall grief in showers;
.+.....+...-....+.-...
Our beauties are not ours. O, I could still,
--...+...-...-....+..+-....+..-..-......+...
Like melting snow upon some craggy hill,
.-....+..-.....+..-.+...-.....+..-..+...
Drop, drop, drop, drop,
..+.....+.....+.....+..
Since nature's pride is now a withered daffodil.
.-.....+.-.......+...-...+..-..+..-.....+..-.+..
```

4 Model

4.1 Input

We used text from English poems as our input data. Each poem contains full punctuation as well as capitalized and lowercase characters. We concatenated all of our poems into one large string while using lines of 50 '*' characters in between the poems to allow the neural network to distinguish between the start and end of poems.
Then we counted the number of different characters in the string in order to perform one-hot encoding of each character so that our string turned into a list of one-hot encodings. We took sequences of length 25 from the list of one-hot encodings and used these sequences as the input to our neural network.

4.2 Output

The network is expected to produce the metrical analysis of the poem character-by-character. For accented syllables it is expected to produce a '+' for the first vowel of the syllable, for unaccented

syllables it is expected to produce a '-' for the first vowel of the syllable, for all other characters it is expected to produce a '.', for new lines it is expected to produce '\n' characters, and in between poems it is expected to produce 50 '*' characters to match the padding we added to our input.
In order to perform supervised learning, we needed to create the output we described above for our large string of poems. We gathered data out of XML files containing each poem with their meter. Since the hyphenator algorithm works only on modern English spellings, we had to do all of the labeling of the poems by hand, spreading out the marks from the meter with '.' characters until they lined up with the first vowel of each syllable. We then again performed one-hot encoding on this output string and passed this to our model as the correct output.
For example, the first line from Shakespeare's Sonnet 18 should have output:

Shall I compare thee to a summer's day?
..-...+..-..+.....-...+.-..+..-.....+..

4.3 Architecture

We used a bidirectional LSTM where each of the left and right LSTMs had 128 output units and a sequence length of 25. Since we had problems with overfitting, we used .3 dropout as well as l2 regularization of the LSTM layers.

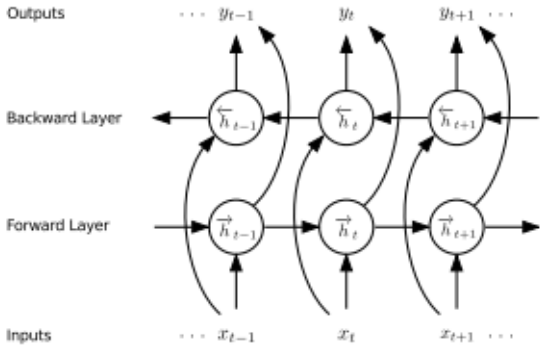

Figure 1: This shows a bidirectional LSTM

5 Experiments And Results

Our experiment was to see how good the bidirectional LSTM was at performing the metrical analysis with it's input being character by character. The following graphs show the data from the bidirectional LSTM using a sequence length of 25, dropout of 0.2, and l2 regularization. Figure 2 is a graph that shows the training and validation loss over epochs. While figure **??** is a graph that shows the training and validation accuracy over epochs. The explanation for why our validation loss increases and the validation accuracy decreases can be found in the discussion section.

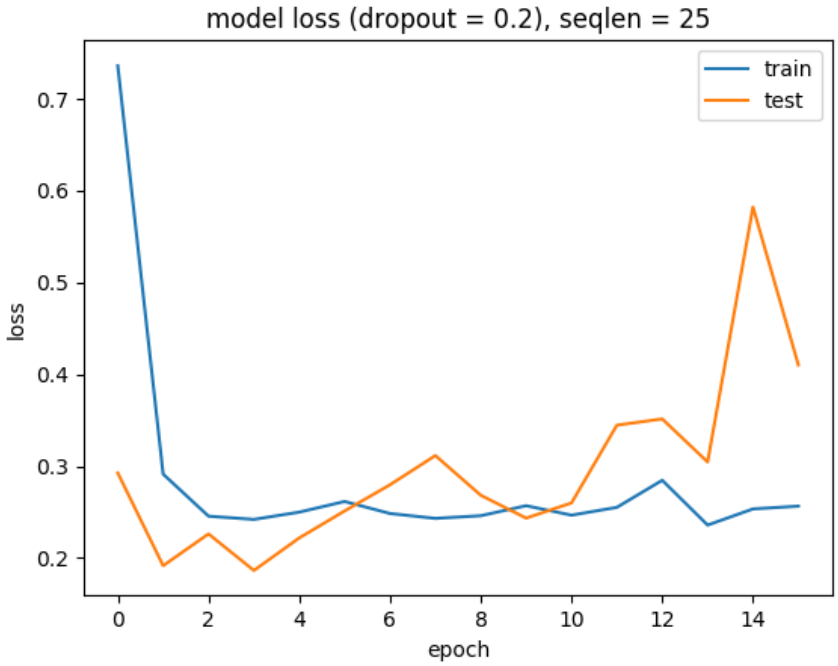

Figure 2: This graph shows the training and validation loss over epochs.

After training the model, we ran a prediction on a single poem used for teaching metrical analysis. Due to the weirdness with the validation results, we tried two different sets of weights when using our model to predict the correct output of the poem. The first set of weights were the weights corresponding to the lowest validation loss (which occurred in epoch 2), and the second set of weights were the weights obtained at the last epoch we ran (epoch 16).

Here is what the correct output should look like:

```
**************************************************
**************************************************
Trochee trips from long to short
..+..-....+.....-...+....-...+..
From long to long in solemn sort
..-...+....-..+...-...+.-....+..
Slow spondee stalks; strong foot yet ill able
..+....+..+....+........+....+....+..+...+..-
Ever to run with the dactyl trisyllable.
+.-...-..+...-.....-..+..-....-.+..-..-.
Iambics march from short to long.
-+..-....+......-....+....-..+...
With a leap and a bound the swift anapests throng.
.-...-..+...-...-..+......-...-...+.-.-.......+...
```

Prediction from weights with lowest validation loss:

```
**************************************************
****************************************
Trochee trips from long to short
..+..-....+.....+...+....-...+..
```

```
From long to long in solemn sort
..+...+....-..+...-...+.-....+..
Slow spondee stalks; strong foot yet ill able
..+....+..-....+........+....+....-..+...+..+.+.-...-..+...+.....-..+.
Ever to run with the dactyl trisyllable.
++..-....+......+....+....-..+...
Iambics march from short to long.
.-...-..+...-...-..+......-...+...-.+.-.......+....
```

Prediction from weights in the last epoch:

```
**********************************************
**************************************
Trochee trips from long to short
..+..-....+.....-...+....-...+..
From long to long in solemn sort
..-...+....-..+...-...+.-....+..
Slow spondee stalks; strong foot yet ill able
..+....+..-....+........+....+....-..+...-..-
Ever to run with the dactyl trisyllable.
+.-...-..+...-.....-..+..-....-.+..-..-.
Iambics march from short to long.
-+..-....+......-....+....-..+...
With a leap and a bound the swift anapests throng.
.-...-..+...-...-..+......-...+...-.+.-.......+...
```

The weights with the lowest validation loss performed poorly in trying to predict the output of the given poem, it did not even have the correct amount of lines. Thus we decided to not trust our validation loss as a correct indicator of how well the weights would do in metrical analysis (the confusing validation loss values are discussed more in the later sections).
Without being able to use the validation loss as valid information, we had to instead use the weights with the lowest training loss which was the last epoch we ran. As we can see from the above results, these weights performed very well when predicting the output of the given poem with only a few errors.
The prediction from weights in the last epoch fail to scan four words correctly: ”spondee”, ”yet”, ”swift”, and ”anapests”. The words ”spondee” and ”anapests” are both not English and therefore have different stress rules so they can be discounted. The neural network labels ”yet” as unstressed and ”swift” as stressed. Out of context, most people would label them such. However, the only reason ”yet” and ”swift” are counted as stressed and unstressed is because each line is following a specific meter. The third line containing ”yet” has a spondaic meter and the line containing ”swift” is anapestic. Since there is no consistent meter throughout the song the only way to know that a line follows that meter is to understand that the author is using the word describing the meter (spondee, dactyl, and anapest) in each of the lines. For this reason, the network confuses ”yet” and ”swift”.

6 Discussion and Conclusions

We think that using bidirectional LSTMs for metrical analysis is promising but our data obtained from validation did not support our claim. At first we thought we were overtraining after only 1 epoch when the validation loss never decreased, but we came to a better conclusion that the main cause was from lack of training data. We were only able to obtain around 80 poems from our poem website, and on top of that some of the poems were short one to four liners. So it makes sense that our validation set, which was created from a 80-20 split of the original data, would have trouble accurately representing the structure/wording of all poems. Our validation loss would rise as we trained our model over many epochs because trying to fit the training data means to have our model learn the metrical analysis of all poems and not learn the small sample size of poems in our validation set which could potentially have little or no relation to the training set.
Regarding the results from our predictions from an already trained network, we performed two experiments with two different sets of weights and we actually found some additional intriguing

results when comparing the outputs from our two experiments. As we mentioned before, the weights corresponding to the lowest validation loss occurred at the beginning epochs, so the outputs of the two experiments can be compared as predictions from a before and after trained model. In the weights with the lowest validation loss representing the weights before our model was trained, the output did not match in terms of number of characters per line and the '+' and '-' symbols stopped matching up with vowels near the end of the poem. However with the weights from the last epoch representing the weights after our model was trained, all of these problems were fixed. The output matched in length with the original poem, and all '+' and '-' symbols lined up with the correct vowels in the corresponding syllables. So we can see that along with learning metrical analysis, our model was also able to learn syllabification and the recognition of certain English characters that occurred within the poems.

7 Concept and Innovation

Our main innovation was in using a bidirectional LSTM for metrical analysis without much preprocessing. In previous work, a lot of manual preprocessing had to be done to syllabify the lines and figure out words. We hoped that this task could have been automated by a hyphenator but these tools do not perform well on the older English and made-up words that appear in these poems. With lack of tools to syllabify the words, we decided on seeing how well our neural network would perform if along with metrical analysis, we let our neural network learn the syllabification of words and recognition of English characters of the poems.
We achieved around 95% accuracy per character. The number of characters per line of our validation set is 38.99. So we achieved around 13% per line accuracy.

Model	Per line (%)
Zeuscansion	26.21
Naive Bayes (10 features)	10.64
Naive Bayes (64 features)	13.88
Perceptron (10 features)	29.32
Perceptron (64 features)	43.36
bidirectional LSTM	13.5

The bidirectional LSTM performed much worse than previous methods. Zeuscansion, which is a rule based system, did twice as good per line and the Perceptron from Agirrezabal et al significantly outperformed it. However, the input to the Perceptron from Agirrezabal required many more features including word, syllable, and part of speech. Agirrezabal does not give any clues as to how they derived most of these features, so we assume there must have been a lot of manual preprocessing. This makes the tool much less useful for performing metrical analysis. Moreover, choosing features creates bias when trying to settle debates on which scansion is correct.
We think that bidirectional LSTMs will be useful to settle debates on different scansions. However, in order for the LSTM to not over fit, a lot more data must be collected. This requires an immense effort of collecting correctly labeled metrical poetry.

References

[1] Agirrezabal, Manex, Inaki Alegria, and Mans Hulden. "Machine Learning for Metrical Analysis of English Poetry." *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (2016): 772-81. Web. 3 Mar. 2017. http://aclweb.org/anthology/C16-1074.

[2] Estes, Alex, and Christopher Hench. "Proceedings of the Fifth Workshop on Computational Linguistics for Literature." *Supervised Machine Learning for Hybrid Meter* (2016): 1-8. Web. 3 Mar. 2017. http://www.aclweb.org/anthology/W16-0201.

[3] Graves, Alex, and Jrgen Schmidhuber. "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures." *Neural Networks* 18.5-6 (2005): 602-10. Web.

8 Individual Contributions

Brandon Ustaris

Brandon helped write the code to produce scansion from a trained network. He also trained multiple variations of the network.

Brian Wilcox

Brian did some of the initial research into models for the problem. He also trained multiple variations of the network as well as research and tweaking (with regularization) to make it perform better.

Brian Preskit

Brian did some research into sparsification to try and make the network run faster.

Derek Tran

Derek helped the write the code to produce scansion from a trained network. He also trained multiple variations of the network.

John Clara

John did the initial research into the problem. He also wrote the code to clean up the data set and make it usable as well as manually labeling the data. He set up the initial version of the network.