Machine Learning for Metrical Analysis of English Poetry

Manex Agirrezabal[1] and **Iñaki Alegria**[1] and **Mans Hulden**[2]

IXA NLP Group[1]   Department of Linguistics[2]
Department of Computer Science   University of Colorado
Univ. of the Basque Country (UPV/EHU)   `mans.hulden@colorado.edu`
`manex.aguirrezabal@ehu.eus`
`i.alegria@ehu.eus`

Abstract

In this work we tackle the challenge of identifying rhythmic patterns in poetry written in English. Although poetry is a literary form that makes use standard meters usually repeated among different authors, we will see in this paper how performing such analyses is a difficult task in machine learning due to the unexpected deviations from such standard patterns. After breaking down some examples of classical poetry, we apply a number of NLP techniques for the scansion of poetry, training and testing our systems against a human-annotated corpus. With these experiments, our purpose is establish a baseline of automatic scansion of poetry using NLP tools in a straightforward manner and to raise awareness of the difficulties of this task.

1 Introduction

Automatic analysis of the rhythmic patterns in poetry may appear deceptively simple. In fact, however, it presents a challenge for structured prediction methods in NLP on par with the most difficult language analysis tasks tackled today. What makes assigning rhythm to written poetry—i.e. "scansion"—a particularly knotty puzzle as a sequence labeling task in NLP is that, while the rhythm in most lines encountered in a work of poetry appears mundanely repetitive on the surface, poetry, while mostly a constrained literary form, is prone to unexpected deviations of such standard patterns. These departures of form, effortlessly understood and analyzed by competent speakers of the language, are tied to multiple levels of language processing. Sometimes, a simple lengthening of a line, the removal of a syllable, an onomatopoetic element, or even a semantic twist to the plot-line in a stanza of poetry can cue a sensitive human reader to assign an apparently deviant rhythmic pattern onto a line of verse.

As an example of the simple and straightforward, consider a line from the ninth book of *Paradise Lost*, by John Milton (Pickering, 1832, p. 128):

No more of talk where God or Angel guest

The even syllables of this line —*more*, *talk*, *God*, *An-* and *guest*—for most readers tend to appear naturally more prominent. At first glance, we might be tempted to assume that this repeats itself through the poem, which indeed is the case.

No more of talk where God or Angel guest
With Man, as with his friend, familiar us'd,
To sit indulgent, and with him partake

However, even here complications arise: in the second line above, we see that the word **with** appears as both unstressed and stressed, showing that the process of assigning prominence to certain syllables cannot depend purely on the lexical items themselves. Still, a naive sequence modeler that simply assumed that the poem follows an unstressed-stressed alternation would fare reasonably well here.

In contrast, consider the first and the fourth quatrains from a well-known poem by Theodore Roethke (1908–1963), *My Papa's Waltz* (1942), which tells the awkward story of a young boy in first-person whose father foists a late-night drunken waltz upon him in the kitchen.

The whiskey on your breath   *You beat time on my head*
Could make a small boy dizzy;   *With a palm caked hard by dirt,*
But I hung on like death: *Then waltzed me off to bed*
Such waltzing was not easy.   *Still clinging to your shirt.*

from Roethke (2011)

The poem, which starts off in the first line with a seeming monotonous regularity of *iambic trimeter*—three syllable-pairs of DE-DUM per line—quickly departs from the form, as if mimicking the angular and erratic 3/4 waltz beats of the drunken father's performance. By the time we reach the word **dizzy** in the second line we find a peculiar extra syllable that needs to be accommodated. But this breaks the pattern, and we now need to decide whether to end the line **small boy di**zzy, or perhaps **small** *boy* **di**zzy? The rhymes also become slanted (as in **dizzy/easy**) perhaps invoking images of slurring, and the natural departure of the DE-DUM rhythmic patterns in what, for most readers, becomes a sequence of two stressed syllables, **beat time** *on my* **head**, conjures up images of the father's whacking the boy in an off-beat fashion.[1]

Most "standard" structured prediction methods effortlessly produce 80%-90% accuracy when assigning levels of stress to syllables, and do so by simply marking the most prominent rhythmic pattern mechanically. One cannot, however, conclude from this that automatic scansion of poetry is a simple task. It merely reflects the pattern of alternation between a large number of regular lines and the unexpected irregular interlude. Moving significantly beyond the accuracies that can be achieved with straightforward machine learning methods remains a challenge for NLP.

In this paper we explore a number of machine learning techniques to automatically assign stress to written poetry against human-annotated gold standards. While we do not expect to be able to tackle highly problematic cases whose solutions require meta-readings, such as understanding the effects of whiskey on the human sense of rhythm, our purpose is to set up a strong baseline and to explore the low-hanging fruits available to us, and to establish the inherent difficulty of the task.

2 Scansion

Conventionally, the metrical scansion of a line of poetry should yield a representation which marks every syllable with its level of stress and divides groups of syllables into units of feet. Typically two or more levels of stress are used. Consider, again, the example line from *Paradise Lost*, whose natural analysis is

```
x   /   x   /    x   /   x   /   x    /
No more |of talk |where god |or An|gel guest
```

where we use the symbol / to denote stressed (ictic) syllables,[2] and x to denote unstressed (non-ictic) ones, as is done in Steele (1999) and the *Princeton Encyclopedia of Poetry and Poetics* (Preminger et al., 2015). The line in question follows the stress pattern

DE-DUM DE-DUM DE-DUM DE-DUM DE-DUM

and consists of five feet of two syllables each with an unstressed-stressed pattern. Indeed, this is the most common meter in English poetry, *iambic pentameter*.

The above example is rather clear-cut. How a particular line of verse *should* be scanned, however, is often a matter of contention. Consider, for example, the first three lines from the poem *Sudden Light* by the English poet Dante Gabriel Rossetti (Rossetti, 1881):[3]

Then, now,–perchance again!
O round mine eyes your tresses shake!
Shall we not lie as we have lain

---

[1]We are lucky to have a recording of Roethke's own rendering of the poem and know that the intended readings of the passages mentioned are **small boy di**zzy and **beat time** *on my* **head**; see `https://www.poets.org/poetsorg/poem/my-papas-waltz-audio-only`.

[2]In the case of inline examples, we will make use of **bold** letters to denote stress.

[3]This stanza did not appear in the first edition of Rossetti's poem book, but did in the second.

The third line is the one that is somewhat ambiguous. The line in question can be read as a sequence of iambs, because the entire poem follows an iambic pattern (*Shall we not lie as we have lain*). But, in a similar manner as is done in the first line from Shakespeare's 18th sonnet (Shakespeare, 1609) (***Shall I** compare thee to a **sum**mers **day***), a commonplace substitution can be made, changing the first iambic foot to a trochee[4] (***Shall** we*). This leads to a so-called *trochaic substitution*. Apart from these two different options, the line could also be analyzed as consisting of two double iambs[5] (*Shall we **not lie** as we **have lain***). Finally, a last possible scansion would be to have assume a trochaic and a pyrrhic foot followed by a double iamb (***Shall** we **not** lie as we **have lain***). In other words, there exists a set of possible analyses for this line which may all be accepted as correct, or at least reasonable.

2.1 State of the art

Automatic scansion of poetry has attracted attention from numerous scholars in recent years and in the following section we discuss some of them. Some works rely on statistical analyses, like Hayward (1996) and Hayes et al. (2012). Others make use of linguistic knowledge obtained by generalizing observations found in different kinds of poetry and propose hand-written rules for the assignment of stress. Recently, as in other NLP tasks, data-driven approaches have emerged in automatic poetry analysis (Estes and Hench, 2016).

Statistics about scansion

Hayward (1996) has as its goal to investigate whether it would be possible to differentiate among the metrical patterns developed by individual writers and also the stylistic differences among periods. To this end, the authors collected a corpus of work by several poets from different time periods and built a neural network model (Rumelhart et al., 1988) to scan poems. Using this technique, Hayward analyzes the work of ten different poets and reports that the neural model of poetic meter was successful in determining "significant differences" among the analyzed poets.

Hayes et al. (2012) propose in their article a new approach to analysis in metrics. Their research is built upon two main works: generative metrics (Halle and Keyser, 1971) and their own earlier work (Hayes and Wilson, 2008) in the use of MaxEnt Grammars for the analysis of phonotactics. They propose a set of constraints that can be assumed to be active when scanning a verse line, and according to the number of times each of these constraints is not fulfilled and according to weights that each constraint have, they determine if a line is metrical or not.

Rule-based scansion

Logan (1988) documents a set of programs to analyze sound and meter in poetry. This work falls in a general genre of techniques that attempt to analyze the phonological structure of poems following the generative phonological theory outlined by Chomsky and Halle (1968) and described by Brogan (1981).

Scandroid is a program that scans English verse written in either iambic or anapestic meter, designed by Charles O. Hartman (Hartman, 1996; Hartman, 2005). The source code is publicly available.[6] The program can analyze poems and check if the predominant stress pattern is iambic or anapestic. However, if the input poem's meter is not one of those two, the system forces each line into one of them. This system represents the current state of the art in the rhythmic analysis of poetry.

AnalysePoems is another tool for identification of metrical patterns written by Plamondon (2006). In contrast with other programs, its main goal is not to perform a perfect scansion, but to only identify the predominant meter in a poem. The program also returns the rhyme scheme that the line follows, such as, **ABCB** for poems whose even lines rhyme.

Calliope is a similar tool, built on top of Scandroid (McAleese, 2007). It is an attempt to leverage syntactic information in order to improve scansion. The program does not appear to be freely available.

---

[4]A sequence of syllables where the first one is stressed and the second one unstressed.
[5]Double iamb: two unstressed syllables and two stressed syllables [`xx//`].
[6]`http://oak.conncoll.edu/cohar/Programs.htm`

One of the recent scansion implementations is *ZeuScansion* (Agirrezabal et al., 2016), a tool for scansion of English poetry, which performs poetry scansion using a simplified version of various stress assignment 'rules-of-thumb' developed by (Groves, 1998). We use this work as a baseline for our experiments.

The rule-based systems mentioned above were designed to work only with poetry in English. There exist, however, several rule-based implementations for other languages, such as Spanish (Gervás, 2000; Navarro-Colorado, 2015).

Supervised Learning & sequential modeling

Estes and Hench (2016) is a current work that makes use of supervised learning tools in order to metrically analyze poems written in Middle High German. Middle High German poetry is a hybrid between qualitative and quantitative verse, which means that both the length and the stress of syllables are taken into account for patterning in the lines. In order to perform supervised learning, they use a corpus of 825 manually annotated lines, which are annotated by the authors. They report an F-score of 0.894 on 10-fold cross-validated development data and 0.904 on held-out testing data.

Unsupervised scansion

Greene et al. (2010) uses statistical methods in the analysis of poetry. For the learning process, *The Sonnets* by Shakespeare was used, as well as a number of other works freely available online.[7] They learn word-stress patterns from the corpus using unsupervised learning and with the incorporation of rhyme and discourse models, they use this system to generate English love poetry. In addition, they also apply their models for the automatic translation of poetry, testing them with Italian three-line stanzas as a source language and English iambic pentameter verse as the target language. We have not obtained an implementation to review.

3   Corpora

As the gold standard material for training our scansion systems, we use a corpus of syllabified and scanned poetry, For Better For Verse (4B4V), from the University of Virginia (Tucker, 2011).[8] This website was originally built as part of an interactive on-line tutorial to train people in the scansion of English poetry in traditional meter. These manually annotated poems can be downloaded from a public repository on GitHub.[9]

The entire collection comprises 78 poems containing approximately 1,100 lines in total. It includes poetry covering a time-span from the 16th century until the 20th and for each century there are at least 6 poems and a maximum of 32 works. Sometimes, several analyses are given as correct in the gold standard to accommodate a natural ambiguity when performing scansion. When two or more analyses are available, we set the error-rate to be the minimum Levenshtein distance to each of the possible analyses, in the same way as in ZeuScansion (Agirrezabal et al., 2016, p. 22) was evaluated.

4   Techniques / Features

We used several Machine Learning algorithms to test our feature configurations, some of them yielding independent outputs, used as a greedy labeler, and some others resulting in structured output. As independent predictors we used an implementation of Naive Bayes (Garner, 1995), Support Vector Machines (Cortes and Vapnik, 1995; Fan et al., 2008) and the averaged Perceptron (Rosenblatt, 1958; Freund and Schapire, 1999).[10] As sequence-based predictors we used the widely employed Hidden Markov Models (Rabiner, 1989; Halácsy et al., 2007) (HMMs) and Conditional Random Fields (Lafferty et al., 2001; Okazaki, 2007) (CRFs).

---

[7]`http://www.sonnets.org`
[8]`http://prosody.lib.virginia.edu/`
[9]`https://github.com/waynegraham/for_better_for_verse/tree/master/poems`
[10]We used an Averaged Perceptron implementation publicly available at `https://bitbucket.org/mhulden/pyperceptron`

Feature template set

Below we show the set of feature templates that our supervised learning scansion systems use, which include:

- Basic features that are (almost) language agnostic

    - Syllable number within the word (SNOW): This specifies the syllable within the word in which we are working currently. E.g., for the word *ha-zel*, whose lexical stress is /x, the specification of the current syllable gives information about the lexical stress of the current syllable.

    - Syllable number within the line (SNL): This feature helps to model the sequence in many types of specially metered lines, e.g., iambic lines. It can resolve possibly ambiguous cases such as the verb re-cord, whose lexical stress could be said to be x/. If we know that this word appears in the last two positions of a trochaic poem, we can ensure that it will have the /x pattern.

    - Number of syllables in the line (NSL): The combination of this feature and the previous one helps in identifying the syllables at the end of a line which are usually more regular because of rhyme patterns.

    - Syllable phonological weight (SWEIGHT): this relies on a generalization that states that heavy syllables—ones which end with a coda consonant or have diphthong nucleus—attract stress. Our hypothesis is that this would be useful in the scansion system, as reflected in John Keats' poem, "*to swe**ll** the **gourd** and **plump** the h**a**z**e**l she**lls***".[11]

    - The last 5 characters of the word (last character, last two characters, last three characters, last four characters, and last five characters) (LC1...LC5): As primary stress of the words in English is usually concentrated in the last syllables of the word (roughly the last three syllables), we expected the last characters to be informative (Hayes, 1995, p. 50). Although it could be better to use the last characters of the syllable, as it was done in Estes and Hench (2016), we tried to be more agnostic about the language in question when developing these basic features, and chose the last characters of the word instead.

    - Word length (WLEN): We expected this to be an informative feature.

- Other features

    - Word: As the main basic units of the text, we used words as features.

    - Syllable: Some syllables are almost always stressed, which could help in the inference of stress patterns. For example, in Shakespeare's Sonnets, the syllable "sire" is used 10 times and in all of them it appears as stressed.

    - POS-tag: The part of speech is a key element to decide whether a word is a content word or function word, which affects the stress in many syllables, as in the following excerpt from *The voice* by Thomas Hardy: "*call to me call to me*", both the verb *call* and the pronoun *me* have lexical stress, but the pronoun loses the prominence when read aloud because it is not a content word. Previous works on poetry analysis, such as Groves (1998), rely on this information.

    - Lexical stress (LS): Knowing the lexical stress sequence in a phrase is an important hint for deducing the rhythmic pattern of a line of poetry. We include the lexical stress of the word that we are analyzing at the moment. This lexical stress is calculated by using the NETTalk dictionary (Sejnowski and Rosenberg, 1987) and when treating out-of-vocabulary words, we calculate their stress using an SVM implementation given in Agirrezabal et al. (2014).

These last four features are extended to include their context as well. For example, we take the current syllable ($syllable[t]$) into account but also additionally its previous and next 10 syllables (syllable[t±10]). In the case of words, part of speech tags, and lexical stresses we decided to include the ± 5 surrounding elements.

---

[11]In this example we use an underline to mark if a syllable is heavy or not.

5 Experiments

We first performed one experiment that only included basic features that could be inferred from each word language-agnostically, and another experiment which included all the features presented above. The simple feature configuration was composed of the first ten features above (SNOW, SNL, NSL, WLEN, SWEIGHT and LC1...LC5). Training greedy sequence predictors with these attributes shows us the basic capability of our predictors using little or (almost) no linguistic information. All these results are compared with the rule-based system ZeuScansion (Agirrezabal et al., 2016), our previous work which we use as our baseline. Results with this feature configuration can be seen in Table 1 and it seems that we could reach quite acceptable (although lower than our baseline) accuracies by simply extracting basic attributes from words. The results of the classifiers using all the features are reported in Table 2. Here, both the SVM and the Perceptron see their scores improve significantly. In the case of the Naive Bayes classifier results do not improve as much as in the other cases, probably because of the sensitivity to overlapping features in Naive Bayes. The difference between the linear SVM and the Perceptron, especially in per-line accuracy, is somewhat noteworthy. Normally, we would expect the SVM, which finds a maximum-margin classification boundary, to outperform the averaged Perceptron, but that is not the case here in both the basic feature set experiment and the full feature set one.

	Per syllable (%)	Per line (%)
Baseline	86.78	26.21
Naive Bayes	78.08	10.64
Linear SVM	83.12	23.40
Perceptron	**84.86**	**29.32**

Table 1: Accuracies of different classifiers using just the basic features (10 features) presented in section 4 using 10-fold Cross-Validation.

	Per syllable (%)	Per line (%)
Baseline	86.78	26.21
Naive Bayes	80.44	13.88
Linear SVM	87.47	35.69
Perceptron	**89.34**	**43.36**

Table 2: Accuracies of different classifiers using all the features (64 features) presented in section 4 using 10-fold Cross-Validation.

From single prediction to structured prediction

As single predictors do not optimize the resulting sequence labeling, they can make simple errors that propagate throughout the line—something that could be avoided by looking at the surrounding outputs. This is the main weakness of not using structured prediction systems.

Hidden Markov Models are simple models that have been successfully used in tasks like POS-tagging, reaching reasonably good results. Conditional Random Fields are often used as an alternative model for POS-tagging and also for Named Entity Recognition and other NLP tasks (McCallum and Li, 2003).

In our experiments, although the per-syllable accuracies do not vary too much, the per-line scores improve substantially by the use of structured predictors. In table 3 the per line and per syllable accuracy of structured prediction systems can be seen (HMM and linear-chain CRF). The HMM has been trained in the standard way, that is, using single syllables (emissions) and their corresponding classes (states). The CRF model is trained analogously, i.e. using only syllables as the features, and the previous label. As expected, training the CRFs using the richer feature configurations employed in the greedy sequence predictors above yields a much higher accuracy, especially in the per line measure. These results are shown in table 4.

	Per syllable (%)	Per line (%)
Baseline	86.78	26.21
Scandroid	89.78	42.95
HMM	**90.43**	**49.88**
CRF	88.13	43.93

Table 3: Accuracy of sequential systems using just syllables.

	CRFs		
	#Features	Per syllable (%)	Per line (%)
Basic features	10	89.66	50.16
All features	64	**91.41**	**55.30**

Table 4: Accuracies of CRFs using different (best) sets of features on 10-Fold Cross-Validation.

6 Discussion & Future work

After checking all the results of the systems, we extracted all the rhythmic pattern predictions of the systems, sorted and grouped them. We can observe that the sorted results of the greedy predictors are more scattered. This is obvious since the greedy predictors do not explicitly promote any holistic coherence at the line level. In the following we show the most common patterns and their frequencies in the same dataset (where roughly 900 lines were used for training and 100 lines were used for testing).

	CRF		**Linear SVM**
26	x / x / x / x / x /	12	x / x / x / x / x /
13	/ x x / x / x / x /	7	/ x x / x / x / x /
5	x / x / x / x / x / x	6	x / x / x / x x x /
5	x / x / x / x /	5	/ / x / x / x / x /
5	x / x /	4	x / x /

In this example it can be seen that given the same dataset, the variance in outputs can be quite different. Both classifiers predict an iambic pentameter most frequently, but, in the case of the Linear Support Vector Machine there are approximately 50 analyses that only appear once (with slight differences between them). On the contrary, the number of unique analyses in the CRF results is around 30. This implicit bias toward regularity is probably helpful for highly regular poetry, but also detrimental for scansion of poetry with often recurring outlier lines. This raises an interesting question that could be tackled in the future: can we include the amount the variation of a single poet as a parameter in the model? Another related question is the amount domain adaptation that can be captured—in this case scanning poetry in a meter that has not been encountered previously.

In this work we have established a baseline for the analysis of rhythmic patterns in poetry using various supervised learning methods. As seen above, there are many cases in which the assignment of stresses is not straightforward which we plan to focus on in future work. We applied typical NLP tools directly and future efforts should focus on the improvement of these techniques, especially in the analysis of lines with syllable additions, removals, ambiguous stress assignments, etc. Additionally, we performed feature selection so as to improve accuracy—this, however, yielded a minimal improvement, but one which was not statistically significant.

The strongest results that we achieved were at around 91.4% per syllable accuracy, averaging 55.3% correctly scanned poetry lines on 10-fold cross-validation (CRF). Comparing with previous approaches to poetry scansion, we outperform rule-based systems such as *Scandroid*—which achieved a 89.78% per syllable and 42.95% per line accuracy—and *ZeuScansion*—which reaches 86.78% per syllable and a 26.21% per line accuracy (Agirrezabal et al., 2016).

Comparing this work with recent results presented in Estes and Hench (2016) on Middle High German poetry, the results that we get in 10-fold cross-validation are quite similar, as they achieve a F-score of .894 on 10-fold cross-validated data and .904 on held-out testing data.

In this article and mainly during our research, we treated the problem as a kind of binary classification task, marking the syllable either as stressed or unstressed. This binarization creates conflicts in some verses in which there is a slow increase of the stress level between syllables. In some works, e.g the above-mentioned Hayes et al. (2012), a four-level stress marking system is used. We believe that doing so can avoid some of the ambiguity problems in scansion. In this sense, the problem could be recast as either a multi-class problem, or a regression problem, calculating a non-binary level of stress for each syllable.

We also expect to improve these results using more advanced techniques. Our first intention is to use current advances in Deep Learning for the analysis of poetry, mostly sequence-based learning paradigms, such as the widely used Recurrent Neural Networks with Long Short Term Memory (LSTM). Our preliminary results with such models show that these frameworks can reach comparable (and sometimes even better) accuracies without extensive work on manual extraction of features.

We have mainly built analyzers for English poetry in this work, but our intention is to investigate if the basic features presented in this work are applicable to poetry written in other languages, and perhaps locate possible typological generalizations among different languages and poetic traditions.

As the corpus of annotated poetry we have used is not very large, we also want to explore the possibility of unsupervised learning of rhythmic patterns in poetry (in a manner similar to Greene et al. (2010)). In this context, the language-agnostic features we have developed should be especially useful. To this end, we plan to learn rhythmic patterns by extracting first the basic, and (nearly) language-universal features by performing basic syllabification according to general principles (Hayes, 2011) such as onset maximization and sonority sequencing.

Acknowledgments

References

Manex Agirrezabal, Jeffrey Heinz, Mans Hulden, and Bertol Arrieta. 2014. Assigning stress to out-of-vocabulary words: three approaches. *International Conference on Artificial Intelligence, Las Vegas, NV*, 27:105–110.

Manex Agirrezabal, Aitzol Astigarraga, Bertol Arrieta, and Mans Hulden. 2016. ZeuScansion: a tool for scansion of English poetry. *Journal of Language Modelling*, 4(1):3–28.

Terry VF Brogan. 1981. *English versification, 1570-1980: a reference guide with a global appendix.* Johns Hopkins University Press.

Noam Chomsky and Morris Halle. 1968. *The sound pattern of English.* Harper & Row, New York.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Alex Estes and Christopher Hench. 2016. Supervised machine learning for hybrid meter. In *Proceedings of the Fifth Workshop on Computational Linguistics for Literature*, pages 1–8. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874. Software available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.

Stephen R. Garner. 1995. Weka: The Waikato environment for knowledge analysis. In *Proceedings of the New Zealand computer science research students conference*, pages 57–64. Citeseer.

Pablo Gervás. 2000. A logic programming application for the analysis of Spanish verse. In *Computational Logic—CL 2000*, pages 1330–1344. Springer.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 524–533. Association for Computational Linguistics.

Peter L Groves. 1998. *Strange music: the metre of the English heroic line*, volume 74. English Literary Studies.

Péter Halácsy, András Kornai, and Csaba Oravecz. 2007. HunPos: an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 209–212. Association for Computational Linguistics.

Morris Halle and Samuel Jay Keyser. 1971. *English stress: Its form, its growth, and its role in verse*. Harper and Row.

Charles O Hartman. 1996. *Virtual muse: experiments in computer poetry*. Wesleyan University Press.

Charles O. Hartman. 2005. The Scandroid 1.1. Software available at http://oak.conncoll.edu/cohar/Programs.htm.

Bruce Hayes and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic inquiry*, 39(3):379–440.

Bruce Hayes, Colin Wilson, and Anne Shisko. 2012. Maxent grammars for the metrics of Shakespeare and Milton. *Language*, 88(4):691–731.

Bruce Hayes. 1995. *Metrical stress theory: Principles and case studies*. University of Chicago Press.

Bruce Hayes. 2011. *Introductory Phonology*. John Wiley & Sons.

Malcolm Hayward. 1996. Analysis of a corpus of poetry by a connectionist model of poetic meter. *Poetics*, 24(1):1–11.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

Harry M Logan. 1988. Computer Analysis of Sound and Meter in Poetry. *College Literature*, pages 19–24.

Gareth McAleese. 2007. *Improving Scansion with Syntax: an Investigation into the Effectiveness of a Syntactic Analysis of Poetry by Computer using Phonological Scansion Theory*. Ph.D. thesis, Open University.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 188–191. Association for Computational Linguistics.

Borja Navarro-Colorado. 2015. A computational linguistic approach to Spanish Golden Age Sonnets: metrical and semantic aspects. *Computational Linguistics for Literature*, page 105.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs). Software available at http://www.chokkan.org/software/crfsuite/.

William Pickering. 1832. *The Poetical Works of John Milton*, volume 2. William Pickering.

Marc R Plamondon. 2006. Virtual verse analysis: Analysing patterns in poetry. *Literary and Linguistic Computing*, 21(suppl 1):127–141.

Alex Preminger, Frank J Warnke, and Osborne Bennett Hardison Jr. 2015. *Princeton encyclopedia of poetry and poetics*. Princeton University Press.

Lawrence R Rabiner. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Theodore Roethke. 2011. *The collected poems of Theodore Roethke*. Anchor.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Dante Gabriel Rossetti. 1881. Poems: A New Edition. *London: Ellis & White*.

David E Rumelhart, James L. McClelland, PDP Research Group, et al. 1988. *Parallel distributed processing*, volume 1. IEEE.

Terrence J. Sejnowski and Charles R. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex systems*, 1(1):145–168.

William Shakespeare. 1609. *Shakespeare's sonnets*. Thomas Thorpe.

Timothy Steele. 1999. *All the fun's in how you say a thing: an explanation of meter and versification*. Ohio University Press Athens.

Herbert F Tucker. 2011. Poetic data and the news from poems: A for better for verse memoir. *Victorian Poetry*, 49(2):267–281.