ZeuScansion: a tool for scansion of English poetry

Manex Agirrezabal, Bertol Arrieta, Aitzol Astigarraga
University of the Basque Country (UPV/EHU)
Dept. of Computer Science
20018 Donostia
manex.aguirrezabal@ehu.es
bertol@ehu.es
aitzol.astigarraga@ehu.es

Mans Hulden
University of Helsinki
Department of modern languages
Helsinki, Finland
mhulden@email.arizona.edu

Abstract

We present a finite state technology based system capable of performing metrical scansion of verse written in English. Scansion is the traditional task of analyzing the lines of a poem, marking the stressed and non-stressed elements, and dividing the line into metrical feet. The system's workflow is composed of several subtasks designed around finite state machines that analyze verse by performing tokenization, part of speech tagging, stress placement, and unknown word stress pattern guessing. The scanner also classifies its input according to the predominant type of metrical foot found. We also present a brief evaluation of the system using a gold standard corpus of human-scanned verse, on which a per-syllable accuracy of 86.78% is reached. The program uses open-source components and is released under the GNU GPL license.

1 Introduction

Scansion is a well-established form of poetry analysis which involves marking the prosodic meter of lines of verse and possibly also dividing the lines into feet. The specific technique and scansion notation may differ from language to language because of phonological differences. Scansion is traditionally done manually by students and scholars of poetry. In the following, we present a finite-state based software tool—ZeuScansion— for performing this task with English poetry, and provide a brief evaluation of its performance on a gold standard corpus of poetry in various meters.

1.1 Scansion

Conventionally, scanning a line of poetry should yield a representation where every syllable is marked with a level of stress—typically two or more levels are used—and groups of syllables are divided into units of feet. Consider, for example, the following line from John Keats' poem *To autumn*.

```
To swell the gourd, and plump the hazel shells
```

Here, a natural analysis is as follows:

```
_   ′    _   ′    _   ′    _   ′  _   ′
To swell |the gourd |and plump |the haz|el shells
```

We use the symbol ′ to denote marked (ictic) syllables, and – to denote unmarked ones (non-ictic). That is, we have analyzed the line in question to follow a stress pattern

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

and also to consist of five feet of two syllables each in the order unstressed-stressed. 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 a line from the poem *Le Monocle de Mon Oncle* by Wallace Stevens:

```
I wish that I might be a thinking stone
```


Proceedings of the 11th International Conference on Finite State Methods and Natural Language Processing, pages 18–24,
St Andrews–Sctotland, July 15–17, 2013.

Here, matters are much more murky. Regarding the ambiguity in this line, the poet Alfred Corn notes that

> ... there is in fact room for disagreement about the scansion of this line. But Stevens is among the most regular of the metrists, and he probably heard it as five iambic feet.[1] Still, an alternative scansion is: one iamb, followed by a pyrrhic foot,[2] followed by two strong stresses, followed by two iambs.

In line with the above commentary, the following represents several alternative analyses of the line in question:

```
Examp.: I wish that I might be a thinking stone

1st:    -  '   -  '  -  '  -  '  -    '

2nd:    -  '   -  -  '  '  -  '  -    '

3rd:    -  '   -  '  '  '  -  '  -    '

4th:    -  '   -  -  -  '  -  '  -    '
```

The first variant is the meter (probably) intended by the author. The second line is Corn's alternative scansion. The third and fourth lines show the output of the software tools Scandroid and ZeuScansion, respectively.

In short, evaluating the output of automatic scansion is somewhat complicated by the possibility of various good interpretations. As we shall see below, when evaluating the scansion task, we use a gold standard that addresses this and accepts several possible outputs as valid.

2 The output of ZeuScansion

As there exist many different established systems of scansion, especially as regards minor details, we have chosen a rather conservative approach, which also lends itself to a fairly mechanical, linguistic rule based implementation. In the system, we distinguish three levels of stress, and mark each line with a stress pattern, as well as make an attempt to analyze the predominant format used in a poem. The following illustrates the analysis produced by our tool of a stanza from Lewis Carroll's poem *Jabberwocky*:

```
1 He took his vorpal sword in hand:
2 Long time the manxome foe he sought-
3 So rested he by the Tumtum tree,
4 And stood awhile in thought.

1 - ' - `- ' - '
2 ' ' - `' ' - '
3 ' `- - - - `- '
4 - ' -' - '
```

Disyllabic feet	
- -	pyrrhus
- '	iamb
' -	trochee
' '	spondee

Trisyllabic feet	
- - -	tribrach
' - -	dactyl
- ' -	amphibrach
- - '	anapest
- ' '	bacchius
' ' -	antibacchius
' - '	cretic
' ' '	molossus

Table 1: Metrical feet used in English poetry

In addition to this, the system also analyzes the different types of feet that make up the line (discussed in more detail below). ZeuScansion supports most of the common types of foot found in English poetry, including *iambs*, *trochees*, *dactyls*, and *anapests*. Table 1 shows a more complete listing of the type of feet supported.

2.1 Metrical patterns

Once we have identified the feet used in a line, we can analyze for each line the most likely meter used. This includes common meters such as:

- Iambic pentameter: Lines composed of 5 iambs, used by *Shakespeare* in his *Sonnets*.
- Dactylic hexameter:[3] Lines composed of 6

[1] Iambic foot: A weak-stressed syllable followed by a strong-stressed syllable.

[2] Pyrrhic foot: Two syllables with weak stress.

[3] Also known as *heroic hexameter*

dactyls, used by *Homer* in the *Iliad*.

- Iambic tetrameter: Lines composed of 4 iambs, used by *Robert Frost* in *Stopping by Woods on a Snowy Evening*.

For example, if we provide Shakespeare's *Sonnets* as input, ZeuScansion classifies the work as *iambic pentameter* in its global analysis (line-by-line output omitted here):

```
Syllable stress _'_'_'_'_'
Meter: Iambic pentameter
```

3 Related work

There exist a number of projects that attempt to automate the scansion of English verse. In this section, we present some of them.

Scandroid (2005) is a program that scans English verse in iambic and anapestic meter, written by Charles O. Hartman (Hartman, 1996). The source code is available.[4] The program can analyze poems and check if their stress pattern is iambic or anapestic. But, if the input poem's meter differs from those two, the system forces each line into iambic or anapestic feet, i.e. it is specifically designed to only scan such poems.

AnalysePoems is another tool for automatic scansion and identification of metrical patterns written by Marc Plamondon (Plamondon, 2006). In contrast to *Scandroid*, *AnalysePoems* only identifies patterns; it does not impose them. The program also checks rhymes found in the input poem. It is reportedly developed in *Visual Basic* and the .NET framwork; however, neither the program nor the code appear to be available.

Calliope is another similar tool, built on top of *Scandroid* by Garrett McAleese (McAleese, 2007). It is an attempt to use linguistic theories of stress assignment in scansion. The program seems to be unavailable.

Of the current efforts, (Greene et al., 2010) appears to be the only one that uses statistical methods in the analysis of poetry. For the learning process, they used sonnets by Shakespeare, as well as a number of others works downloaded from the Internet.[5] Weighted finite-state transducers were used for stress assignment. As with the other documented projects, we have not found an implementation to review.

[4] http://oak.conncoll.edu/cohar/Programs.htm

[5] http://www.sonnets.org

4 Method

Our tool is largely built around a number of rules regarding scansion developed by Peter L. Groves (Groves, 1998). It consists of two main components:

(a) An implementation of Groves' rules of scansion—mainly a collection of POS-based stress-assignment rules.

(b) A pronunciation lexicon together with an out-of-vocabulary word guesser.

(a) Groves' rules

Groves' rules assign stress as follows:

1. Primary step: Mark the stress of the primarily stressed syllable in content words.[6]
2. Secondary step: Mark the stress of (1) the secondarily stressed syllables of polysyllabic content words and (2) the most strongly stressed syllable in polysyllabic function words.[7]

In section 5 we present a more elaborate example to illustrate how Groves' rules are implemented.

(b) Pronunciation lexicon

To calculate the basic stress pattern of words necessary for step 1, we primarily use two pronunciation dictionaries: The CMU Pronouncing Dictionary (Weide, 1998) and NETtalk (Sejnowski and Rosenberg, 1987). Each employs a slightly different notation, but they are similar in content: they both mark three levels of stress, and contain pronunciations and stress assignments:

```
NETTALK format:
abdication     @bdIkeS-xn     2<>0>1>0<<0

CMU format:
INSPIRATION    IH2 N S P ER0 EY1 SH AH0 N
```

The system uses primarily the smaller NETtalk dictionary (20,000 words) and falls back to use CMU (125,000 words) in case a word is not found

[6] Content words are nouns, verbs, adjectives, and adverbs.

[7] Function words are auxiliaries, conjunctions, pronouns, and prepositions.

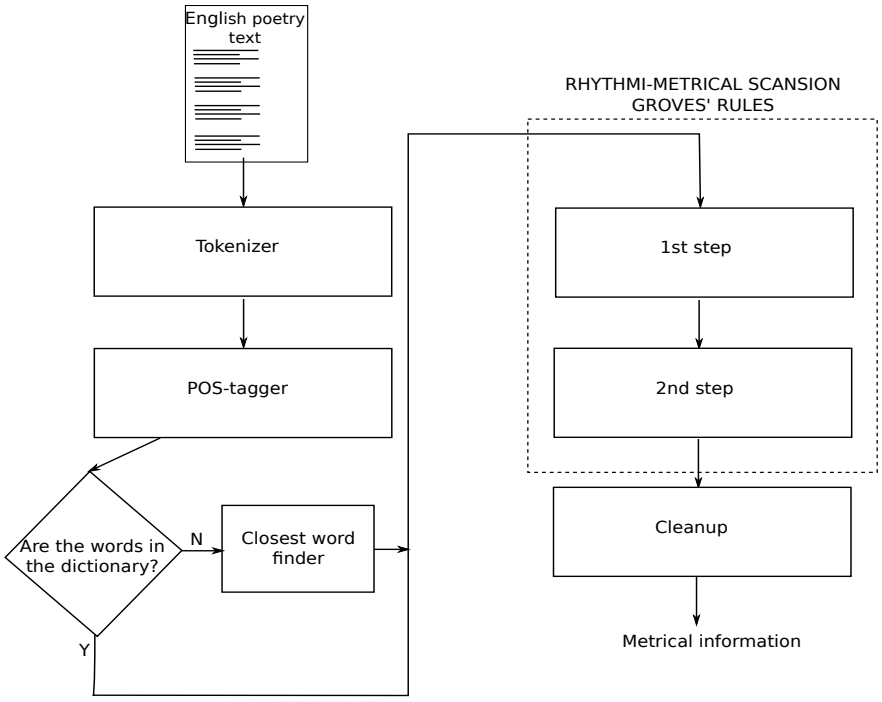

Figure 1: Structure of ZeuScansion

in NETtalk. The merged lexicon, where NETtalk pronunciations are given priority, contains some 133,000 words.

5 *ZeuScansion*: Technical details

The structure of the system is divided into the subtasks shown in figure 1. We begin with preprocessing and tokenization, after which words are part-of-speech tagged. Following that, we find the default stresses for each word, guessing the stress patterns if words are not found in the dictionary. After these preliminaries, we apply the steps of Groves' scansion rules and perform some cleanup of the result.

The toolchain itself is implented as a chain of finite-state transducers using the *foma*[8] toolkit (Hulden, 2009), save for the part-of-speech tagger which is a Hidden Markov Model (HMM) implementation (Halácsy et al., 2007). We use *Perl* as a glue language to communicate between the components.

[8]http://foma.googlecode.com

Preparation of the corpus

After tokenization,[9] we obtain the part of speech tags of the words of the poem. For the POS-tagger, we trained Hunpos[10] (Halácsy et al., 2007) with the Wall Street Journal English corpus (Marcus et al., 1993). While other more general corpora might be more suitable for this task, we only need to distinguish between function and non-function words, and thus performance differences would most likely be slight.

Once the first process is completed, the system starts applying Groves' rules, which we have encoded as finite-state transducers. To apply the rules, however, we must know the stress pattern of each word. The main problem when assigning patterns is that the pronunciation of some words will be unknown, even though the dictionaries used are quite large. This often occurs because a word is either misspelled, or because the poem is old and uses archaic vocabulary or spellings.

The strategy we used to analyze such words was to find a 'close' neighboring word in the dictionary, relying on an intuition that words that differ very lit-

[9]https://code.google.com/p/foma/wiki/FAQ

[10]https://code.google.com/p/hunpos

tle in spelling from the sought-after word are also likely pronounced the same way.

Finding the closest word

In order to find what we call the 'closest word' in the dictionary, we construct a finite-state transducer from the existing dictionaries in such a way that it will output the most similar word, according to spelling, using a metric of word distances that we have devised for the purpose. Among other things, the metric assigns a higher cost to character changes toward the end of the word than to those in the beginning (which reflect the onset of the first syllable), and also a higher cost to vowel changes. Naturally, fewer changes overall also result in a lower cost. For example, in the following line from Shakespeare's *Romeo and Juliet*:

*And **usest** none in that true use indeed*

we find the word **usest**, which does not appear in our lexicon.[11] Indeed, for this word, we need to make quite a few changes in order to find a good 'close' match: **wisest**.

Groves' rules

Once we have obtained the stress pattern for each word, we begin to apply Groves' rules: to stress the primarily stressed syllable in content words. This is implemented with a finite state transducer built from replacement rules (Beesley and Karttunen, 2003) that encode the steps in the rules. In our *Hamlet* example, for instance, our input to this stage looks like this:

```
to+-+TO be+'+VB or+-+CC not+'+RB to+-+TO ...
 that+-+IN is+-+VBZ the+-+DT question+'-+NN

   the+-+DT uncertain+-'-+JJ sickly+`-+JJ
     appetite+'--+NN to+-+TO please+'+VB
```

Next, we apply the second rule—that is, we mark the secondarily stressed syllables of polysyllabic content words and the most strongly stressed syllable in polysyllabic function words:

```
to+-+TO be+'+VB or+-+CC not+'+RB to+-+TO ...
 that+-+IN is+-+VBZ the+-+DT question+'-+NN

   the+-+DT uncertain+`'-+JJ sickly+`-+JJ
     appetite+'-`+NN to+-+TO please+'+VB
```

[11] The archaic second-person singular simple present form of the verb **use**.

The last step is to remove all the material not needed to work with stress patterns. In this part, we get as input a sequence of tokens with a specified structure:

```
inspiration+`-'-+NN
```

For the cleanup process, we use a transducer that removes everything before the first + character and everything after the second + character. It next removes all the + characters, so that the only result we get is the stress structure of the input word.

Global analysis

After the stress rules are applied, we attempt to divide lines into feet in order to produce a global analysis of the poem. Since foot-division can be ambiguous, this is somewhat non-trivial. Consider, for instance, the meter:

```
_' __' __' __' _
```

which could be analyzed as consisting mainly of (1) amphibrachs [-'-], (2) trochees ['-] and (3) iambs [-']. All three patterns appear four times in the line. For such cases, we have elaborated a scoring system for selecting the appropriate pattern: we give a weight of 1.0 for hypothetical disyllabic patterns, and a weight of 1.5 for trisyllabic ones. In this example, this would produce the judgement that the structure is amphibrachic tetrameter (1.5 $\times$ 4 matches = 6).

Foot	Pattern	N^o matches	Score
Amphibrach	–' –	4	6
Iamb	–'	4	4
Trochee	' –	4	4
Anapest	'–	3	4.5
Dactyl	'–	3	4.5
Pyrrhus	- -	3	3

6 Evaluation

As the gold standard material for evaluation, we used a corpus of scanned poetry, *For Better For Verse*, from the University of Virginia.[12] We extracted the reference analyses from this website, which originally was built as an interactive on-line tutorial to train people in the scansion of English poetry in traditional meter. Sometimes several analyses

[12] http://prosody.lib.virginia.edu

	Scanned lines	Correctly scanned
No CWF	759	173
With CWF	759	199
No CWF	Accuracy: 22.79%	
With CWF	Accuracy: 26.21%	

	Scanned sylls.	Correctly scanned
No CWF	7076	5802
With CWF	7076	5999
No CWF	Accuracy: 81.995%	
With CWF	Accuracy: 86.78%	

Table 2: ZeuScansion evaluation results against the *For better or Verse* corpus. The CWF label indicates whether the closest word finder was used for assigning stress to unknown words.

are given as correct. The results of the evaluation are given in table 2. As seen, 86.78% of syllables are scanned correctly in the best configuration. We include scores produced without the word guesser component to show its significance in the process.

For checking the number of correctly scanned syllables on each line, we use *Levenshtein* distance in comparing against the gold standard. We do this in order not to penalize a missing or superfluous syllable—which are sometimes present—with more than 1 count. For example, the two readings of Stevens' poem mentioned in the introduction would be encoded in the corpus as

−+−+−+−+−+|−+−−++−+−+

while our tool marks the line in question as

−+−−−+−+−+

after conversion to using only two levels of stress from the original three-level marking. Here, the minimum *Levenshtein* distance between the analysis and the reference is one, since changing one − to a + in the analysis would equal the first 'correct' possibility in the gold standard.

Closest word finder

Since the closest word finder has some impact on the overall quality of the system, we have evaluated that component separately. Figure 2 shows a graph illustrating the increasing coverage of words depending on distance to the neighboring word used as a pronunciation guide for out-of-vocabulary items. The first column of the graph (NC) represents the percentage of the corpus that could be read using only the dictionaries, while in the following ones, we show the improvements we get in terms of coverage using various substitutions. The codes that appear in the lower part of figure 2 refer to the allowed changes. The first letter can be either `B` or `E`. If it is `B`, the changes will be made in the beginning of the word. The character following the hyphen describes the changes we allow subsequently: for example, `VC` corresponds to the change of one vowel and one consonant.

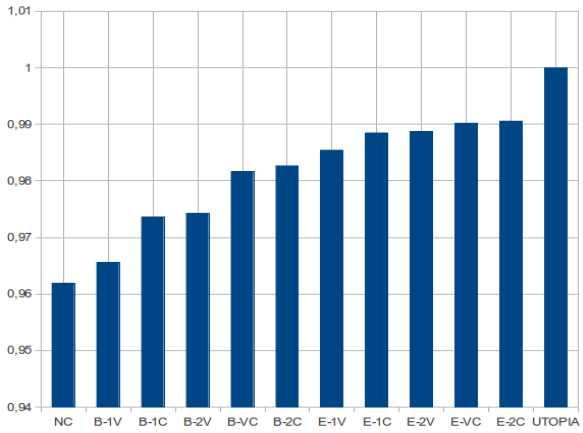

Figure 2: Evaluation of the closest word finder

7 Discussion and future work

In this work, we have presented a basic system for scansion of English poetry. The evaluation results are promising: a qualitative analysis of the remaining errors reveals that the system, while still containing errors vis-à-vis human expert judgements, makes very few egregious errors. The assignment of global meter to entire poems is also very robust. We expect to develop the system in several respects. Of primary concern is to add statistical information about the global properties of poems to resolve uncertain cases in a manner consistent with the overall structure of a given poem. Such additions could resolve ambiguous lines and try to make them fit the global pattern of a poem. Secondly, there is

still room for improvement in unknown word performance. Also, the part-of-speech tagging process may be profitably replaced by a deterministic FST-based tagger such as Brill's tagger, as presented in Roche and Schabes (1995). This would allow the representation of the entire tool as a single FST.

We believe that the availability of a gold-standard corpus of expert scansion offers a valuable improvement in the quantitative assessment of the performance of future systems and modifications.

Acknowledgments

We must refer to Herbert Tucker, author of the "For Better for Verse" project, which has been fundamental to evaluate our system. We also must mention the Scholar's Lab, an arm of the University of Virginia Library, without whose aid in development and ongoing maintenance support the cited project would be impossible. Joseph Gilbert and Bethany Nowviskie have been most steadily helpful there.

References

Beesley, K. R. and Karttunen, L. (2003). Finite-state morphology: Xerox tools and techniques. *CSLI, Stanford*.

Carroll, L. (2003). *Alice's adventures in wonderland and through the looking glass*. Penguin.

Fussell, P. (1965). *Poetic Meter and Poetic Form*. McGraw Hill.

Greene, E., Bodrumlu, T., and Knight, K. (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.

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

Halácsy, P., Kornai, A., and Oravecz, C. (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.

Hartman, C. O. (1996). *Virtual muse: experiments in computer poetry*. Wesleyan University Press.

Hulden, M. (2009). Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pages 29–32. Association for Computational Linguistics.

Keats, J. (2007). *Poems published in 1820*. Project Gutenberg.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

McAleese, G. (2007). Improving scansion with syntax: an investigation into the effectiveness of a syntactic analysis of poetry by computer using phonological scansion theory. -.

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

Roche, E. and Schabes, Y. (1995). Deterministic part-of-speech tagging with finite-state transducers. *Computational linguistics*, 21(2):227–253.

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

Shakespeare, W. (1609). *Shakespeare's sonnets*. Thomas Thorpe.

Shakespeare, W. (1997). *Romeo and Juliet*. Project Gutenberg.

Shakespeare, W. (2000). *The tragedy of Hamlet*, volume 1122. Project Gutenberg.

Stevens, W. (1923). *Harmonium*. Academy of American Poets.

Weide, R. (1998). The CMU pronunciation dictionary, release 0.6.