Assigning stress to out-of-vocabulary words: three approaches

Manex Agirrezabal*, Jeffrey Heinz†, Mans Hulden§ and Bertol Arrieta*
*University of the Basque Country (UPV/EHU)
IXA NLP Group, Department of Computer Science
Donostia, Basque Country 20018
†University of Delaware
Department of Linguistics & Cognitive Science
Newark, Delaware 19716 (USA)
§University of Colorado Boulder
Department of Linguistics
Boulder, Colorado (USA)
manex.aguirrezabal@ehu.es, heinz@udel.edu, mans.hulden@colorado.edu, bertol.arrieta@ehu.es

***Abstract*—In this paper we address the task of automatically assigning primary stress to out-of-vocabulary words in English. This work forms a necessary component in a scansion system for English poetry. We propose three different approaches based on (1) word similarity, (2) hand-written linguistic rules and (3) machine learning. The first and last approach require stress-annotated corpora to train a model for stress assignment, while the linguistic approach relies on grapheme to phoneme conversion, a syllabification procedure and hand-written stress assignment rules. An implementation of each approach is provided and the precision of the systems is compared. The linguistic approach proves to be the most effective, but the machine learning approach is not far behind. We anticipate that including part of speech information will improve the accuracy of each system. The source code of the approaches is released[1] under the GNU GPL license.**

I. Introduction

Automatic scansion of English Poetry has attracted different scholars from all around the world. *Scandroid* [1], *AnalysePoems* [2] and *Calliope* [3] are some examples of computational systems created to analyze English verse. [4] is another recent attempt to scan English poetry using unsupervised learning algorithms and weighted finite-state machines.

The authors would like to acknowledge the University of Delaware, the Department of Linguistics and Cognitive Science in particular, for receiving the first author as a visiting student. Also we would like to thank the University of the Basque Country, whose grant has made possible this stay in Newark, Delaware.

[1]Code available at http://athenarhythm.googlecode.com

The purpose of a scansion system is to determine the rhythmic nature of lines of verse. It has long been recognized that knowing the location of primary stress in English words is an important component of successful scansion of English poetry. Typically, this information can be found in dictionaries. However, many poems make use of words which are *not* found in *any* dictionary. This is because these words have atypical spellings, are derived through a series of complex morphological processes, or because they are nonce words completely made up (cf. Lewis Carroll's 'the vorpal blade'). The problem of out-of-vocabulary words is greater for older English poetry. Scansion systems which cannot determine the location of primary stress in such out-of-dictionary words introduce a source of error wherever they occur. The scansion of a poetry line gives a representation where each syllable is marked with a level of stress and these syllables are then grouped into feet. Here you can see an example from Henry Wadsworth Longfellow's *The song of Hiawatha* and it's scansion, where stressed syllables are marked with the symbol ′:

```
′     - |  ′   - |  ′  -| ′  -
And the | mighty | Mudje|keewis,
```

This paper examines three systems which aim to reduce this sort of error by correctly predicting the location of stress in out-of-dictionary words. The first approach is based on *analogy*: a word *similar* to the target word is searched in a dictionary. The *linguistic* approach follows from the observation by linguists that, in many

of the world's languages, the location of primary stress in words is largely predictable [5], [6], and may often be encoded as phonological generalizations, or rules. These systems of rules or constraints can be used to predict the location of primary stress on words not found in dictionaries. The third approach is based on *machine learning*. A dictionary can be used to train a system to predict the location of primary stress on words not found in dictionaries. This is not unlike the linguistic approach. Both the linguist and the machine are inferring patterns from data. The difference is in the kinds of models the two approaches entertain and whether the inference is made mentally or automatically.

We emphasize that there are several ways each of the above approaches can be instantiated. For analogy-based systems, for example, there are many different solutions to the problem of defining 'word similarity.' Also, linguists disagree to some extent about the nature of the linguistic generalizations regarding stress assignment in English (see for example [7]). Likewise, there are many machine learning techniques that could in principle be applied to the task of predicting the location of stress in words [8]. It is of course beyond the scope of this paper to investigate all the possibilities. However, the implementation choices made here straightforwardly represent each of these approaches.

Our results show that both the linguistic and machine learning approaches significantly outperform the analogical approach. Additionally, the linguistic approach slightly outperforms the machine learning approach.

II. Background

The background for this work is found in ZeuScansion [9], a tool for scansion of English poetry based on finite-state technology. This system was designed around several finite-state machines that analyzed verse by performing tokenization, part of speech tagging, stress placement, and—for unknown words—stress pattern guessing. The analogy method examined here is the one used originally in the ZeuScansion tool.

There exist various linguistic approaches for stress allocation, such as the one proposed in [5]. The works [6] and [7] also address the issue of stress location from a linguistic point of view. This paper employs a slightly simplified version of the rules mentioned in these references as generalization rules.

Only a few studies have investigated machine learning of stress systems from corpora.[2] These include [12], where Russian stress location in predicted using Maximum Entropy Ranking and [13] which uses a Support Vector Machine (SVM) ranker with linguistically-inspired features. Our machine-learning method is most similar to Dou et al. [13], though we employ different features. They operate with more linguistically motivated features, while our system works with more standard ones. Based on their reported results, we would expect their methods to outperform our machine learning method. However, we do not have a direct comparison and leave such a comparison for future research.

III. Corpus

We have used the NETtalk pronunciation dictionary [14] for training and testing our models. The dictionary contains various pieces of information about English words such as: pronunciation, number of syllables, possible part of speech and primary and subsidiary stress location. The version we have employed uses an ASCII encoding for the phonemic representations of words.

IV. Task

As mentioned before, the main task to be resolved in this paper is the main stress assignment of unknown English words. In this section, we describe three approaches that perform it.

A. Similarity approach

The similarity approach was first proposed in [9]. The system, represented as a finite-state machine [15], attempts to make small changes to the unknown word to yield a similar known word. We suppose that the scansion of the 'most similar' word found in the dictionary will be the same as that of the input word. For example, in Lewis Carroll's *Phantasmagoria and Other Poems* the following phrase can be found, whose scansion cannot be inferred by simply using dictionaries:

I prophesy there'll be a row

The word "prophesy" is not found in our dictionary, so the similarity approach can find out that the most similar word is "prophecy", whose first syllable will receive main stress.

[2] A few studies with a cross-linguistic emphasis examine learning from artificial corpora [10], [11].

The framework could be described as a series of transducers. The idea is that each transducer is programmed to make specific changes in specific parts of a word. The allowed changes are substitution, deletion and addition of one or two letters (a consonant or a vowel) and we divide the word in two different parts: The last part roughly corresponds to the final syllable and the first part to the rest of the word. For example, the words "deletion" and "smartphone" would be divided as "dele|tion" and "smartp|hone".

These are the changes that can be done using the transducers of the machine, which can be applied either in the first or the second part of the word.

- Change one vowel
- Change one consonant
- Change two vowels
- Change one vowel and a consonant
- Change two consonants

In total, we will have ten transducers for performing the above operations[3]. Each of them will produce an output if a word in the dictionary is found by making the allowed changes to the input word. These blocks will be installed as an ordered set of transducers, and if the first transducer gives an output, this will be returned by the entire system. If the first transducer can't give an output, the same input will be given to the second transducer, and so forth. That is, the order of these transducers will reflect a priority that certain changes have over other changes. The output will be preferentially the one given by the top transducers.

It is noteworthy that no matter the order of the transducers is, the recall of the entire system will not vary. However, the precision will change, so in order to find the best ordering we have tried to optimize the precision. To this end, we calculated the precision of each of the ten transducers and decided to place first the one that produced the highest precision, etc.

B. Linguistic approach

In this approach we have programmed a linguistic chain that performs grapheme to phoneme conversion, syllabification and stress assignment.

[3]It's important to mention that when a character is replaced, an element of the same category will be inserted. So, for example, if we change a vowel, we can't put a consonant in it's place, only a vowel.

As mentioned in [16], we assume that English words first have to be syllabified after which word stress rules are applied, which in turn depend on the syllable type. Before syllabification we perform a grapheme-to-phoneme conversion, based on [17]. The `g2p` program uses a model trained from the NETtalk dictionary for the task. After this, we syllabify the word with the finite-state syllabification system presented in [18]. The main concern for the stress assignment is the weight of the syllables, which might be light or heavy:

Heavy syllable: The syllable has a coda or ends in a tense vowel.

Light syllable: The other set of syllables will be considered light.

Once we apply all these processes to the input word, we execute some main stress assignment rules, encoded as a set of finite-state transductions. The main active rule is the so-called Latin stress rule ([6], P.50), which also applies in many English polysyllabic words. The rule in question codifies the generalization that heavy syllables tend to attract stress. Below is a description of this generalization, divided into four subrules:

- If the penultimate syllable is light, the antepenultimate syllable is stressed
- If the penultimate syllable is heavy, it is stressed
- In the case of disyllabic words, the first syllable is stressed
- Monosyllabic words are stressed

In figure 1 you can see the workflow of the linguistic chain with an example. Anyway, there are words that don't follow these generalization rules, such as, "an.té.nna", "A.la.bá.ma" or "po.líce". Hayes [6] estimates that in the 87% of English vocabulary, main stress can be predicted using linguistic rules. Using the aforementioned rules, it can be seen in the results section that we have not reached that precision.

C. Machine Learning approach

For the third approach we have trained Support Vector Machines [20] [21] using a stress-annotated dictionary. We have treated the stress assignment task as a multi-class classification problem. In the task, the class to be assigned is the stress pattern that each word follows, taking into account only the main stress. We extracted 25 different stress patterns from our corpus.

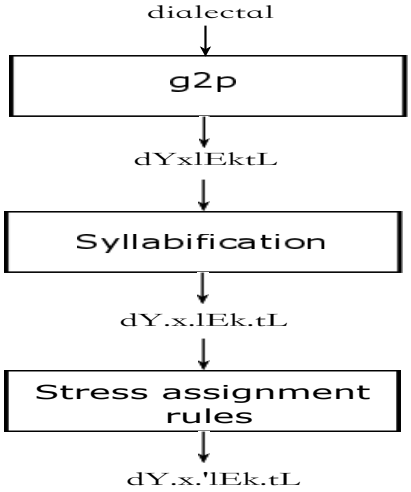

Fig. 1. Workflow of the linguistic approach.

We extracted two different sets of features for the purpose of training the SVMs. In the first set, *FS1*, we used character bigrams as features, including word boundaries. In the second feature set, *FS2*, we used character trigram frequencies, also known as Wickelfeatures [22]. For example, given the word "abate", in *FS1* we would train the SVM with the information that the {#a}, {ab}, {ba}, {at}, {te}, {e#} appeared once and the other possible bigrams zero times. These, together with the length of the word, are the training features for the first set. In the second model, we include the frequencies of trigrams {#ab}, {aba}, {bat}, {ate}, {te#}. For the example word "abate", the correct class would be $< \_' >$, indicating that the second syllable carries primary stress. Naturally, the features need to be encoded as numbers, so we used a dictionary and a mapping function[4] for the first feature set:

$$D(ch) = \{\#\} \rightarrow 0, \{a\} \rightarrow 1, \{b\} \rightarrow 2, ...$$
$$map(ch0, ch1) = ch0 * L^0 + ch1 * L^1$$

In the case of the second feature set the mapping function is similar but with some improvements to reduce the number of features.

After converting the data, we produce a corpus with 19,528 instances, one instance per word in the original corpus. In FS1, each instance has 899 attributes. On the other hand, in FS2, each instance has 5,495 attributes.

We trained different Support Vector Machines and evaluated each of them using 10-fold cross-validation.

[4] L will be the length of the dictionary D

In the next table, you can see the accuracy that we can get using different linear solvers[5] with each feature set. In some cases we have not included the accuracy because the computer ran out of memory.

Linear SVMs		
Solver type	Accuracy	
	FS1	FS2
1	0.48177	-
2	0.30065	-
3	0.41991	0.44659
4	0.50195	0.53899
5	**0.56580**	-

We also used nonlinear kernels[6] in our experiments. In the next table you can see the different results we got for each feature set.

Non-linear SVMs		
Kernel type	Accuracy	
	FS1	FS2
P	0.28052	0.278626
RBF	**0.461594**	0.278626
S	0.431176	0.278626

After training different SVMs and kernels, we performed a grid-search as it is recommended in [23] trying different C and γ parameters for the best SVMs, kernels and feature sets. Finally, the highest accuracy of 70.9820% was found by using the C-Support Vector Classifier with a RBF kernel and $C = 1024$ and $\gamma = 0.0078125$.

V. Experiment and results

In this section, we present the results and evaluation of the previously mentioned systems. In the following table, you can see where each of the approaches assign primary stress to these out-of-vocabulary words:

[5] 1: L2-regularized L2-loss SVC (dual)
2: L2-regularized L2-loss SVC (primal)
3: L2-regularized L1-loss SVC (dual)
4: SVC by Crammer and Singer
5: L1-regularized L2-loss SVC

[6] P: Polynomial
RBF: Radial Basis Function
S: Sigmoid

Input word	SIM-OO	LING	SVM
mudjekeewis	X	__'_	'__
pantroductory	__'__	__'__	__'__
joology	_'__	_'__	'_
amtenna	_'_	'__	'__
cate	'_	'	'

As some of the above methods are data-driven and others not, we evaluated each one slightly differently. Both the similarity approach and the machine learning approach were evaluated using 10-fold cross-validation. The linguistic approach, however, was evaluated with the whole corpus without any splitting, as it does not rely on any training data and is essentially an expert system.

In the following table[7] the obtained results can be observed:

	Accuracy
SIM	0.5843
SIM-OO	0.6777
LING	**0.7362**
SVM	0.7098

At this point, we can tentatively establish that the best result is obtained using the linguistic generalizations. However, the accuracies of both SVMs and hand-encoded generalizations are sufficiently close to warrant further research in improvement of both linguist-devised rules on the one hand, and better use of features for SVM modeling on the other.

VI. Discussion & Future Work

In this work, we have presented three different approaches to assigning a stress pattern to English out-of-vocabulary words. Accurate stress assignment is an important component of many larger systems, such as a scansion system discussed above.

The three approaches that we have proposed and evaluated are based on (1) word similarity, (2) linguistic rules and (3) machine learning. The results show that the best performance can be got using the linguistic generalizations, with a 73.62% precision, or the Support Vector Machine, with 70.98%.

There are several ways we believe the system's performance can be improved. One way is to include part-of-speech information. Other method, for the machine learning approach, is to experiment with different feature sets. In the case of the linguistic approach, we think that the rules can still be polished.

We are aware that trying to guess English word stress without having the information about the part of speech, can result in several errors. For example, in words like "present" primary stress can be placed both in the first or the second syllable, depending on the part of speech[8]. Having access to part of speech information in the linguistic and machine learning paradigm will without doubt improve the results of each one, perhaps significantly.

Furthermore, we hope that adding more features to the SVM and applying a feature selection algorithm will improve its results. In this regard, the work by Dou et al. is likely to be instructive. The features employed in that particular work are substrings that are slightly different from the n-grams employed here. Their feature-substrings consist of all the vowels in a word, with maximally one surrounding consonant include on both sides (if present), mimicking syllables in a way.

Additionally, in the SVM approach, we are considering to improve the feature mapping function, trying to map the proximity between letters[9], based on the IPA table.

For the linguistic approach, we are considering improving the rules, especially the one referring to the disyllables, in which stress is always located in the first syllable. Our insight is that primary stress can be assigned taking into account the weight difference between the two syllables.

Another future extension of these algorithms is to develop them for other languages. In this case, the machine learning approach is especially interesting because no linguistic knowledge is necessary, only a training corpus of words marked with stress.

References

[1] C. O. Hartman, *Virtual muse: experiments in computer poetry*. Wesleyan University Press, 1996.

[2] M. R. Plamondon, "Virtual verse analysis: Analysing patterns in poetry," *Literary and Linguistic Computing*, vol. 21, no. suppl 1, pp. 127–141, 2006.

[7] SIM: Similarity approach
SIM-OO Similarity approach w/ optimal ordering
LING: Linguistic approach
SVM: Support Vector Machine (with parameter C=30)

[8] If the word is a noun, the stress goes on the first syllable (présent). If it is a verb, the second syllable is stressed (presént).

[9] The distance between 'p' and 'b' would be smaller than the one between 'b' and 'k'

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

[4] E. Greene, T. Bodrumlu, and K. Knight, "Automatic analysis of rhythmic poetry with applications to generation and translation," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2010, pp. 524–533.

[5] M. Halle and J.-R. Vergnaud, *An essay on stress*. Mit Press Cambridge, 1987.

[6] B. Hayes, *Metrical stress theory: Principles and case studies*. University of Chicago Press, 1995.

[7] L. Burzio, *Principles of English stress*. Cambridge University Press, 1994, no. 72.

[8] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[9] M. Agirrezabal, B. Arrieta, A. Astigarraga, and M. Hulden, "ZeuScansion: a tool for scansion of English poetry," *Finite State Methods and Natural Language Processing*, p. 18, 2013.

[10] E. Dresher and J. Kaye, "A computational learning model for metrical phonology," *Cognition*, vol. 34, pp. 137–195, 1990.

[11] J. Heinz, "On the role of locality in learning stress patterns," *Phonology*, vol. 26, no. 2, pp. 303–351, 2009.

[12] K. Hall and R. Sproat, "Russian stress prediction using maximum entropy ranking," *Empirical Methods in Natural Language Processing*, 2013.

[13] Q. Dou, S. Bergsma, S. Jiampojamarn, and G. Kondrak, "A ranking approach to stress prediction for letter-to-phoneme conversion," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, ser. ACL '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 118–126.

[14] T. J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce English text," *Complex systems*, vol. 1, no. 1, pp. 145–168, 1987.

[15] M. Hulden, "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*. Association for Computational Linguistics, 2009, pp. 29–32.

[16] S. Duanmu, H.-Y. Kim, and N. Stiennon, "Stress and syllable structure in English: Approaches to phonological variations," *University of Michigan*, 2005.

[17] J. Novak, N. Minematsu, and K. Hirose, "Wfst-based Grapheme-to-Phoneme conversion: Open source tools for alignment, model-building and decoding," *Finite-State Methods and Natural Language Processing*, 2012.

[18] M. Hulden, "Finite-state syllabification," *Finite-State Methods and Natural Language Processing*, pp. 86–96, 2006.

[19] R. A. Mester, "The quantitative trochee in latin," *Natural Language & Linguistic Theory*, vol. 12, no. 1, pp. 1–61, 1994.

[20] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[21] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[22] D. E. Rumelhart and J. L. McClelland, *On learning the past tenses of English verbs*. Institute for Cognitive Science, University of California, San Diego, 1985.

[23] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, "A practical guide to support vector classification," 2003.

[24] L. Carroll, *Alice's adventures in wonderland and through the looking glass*. Penguin, 2003.

[25] H. W. Longfellow and H. R. Schoolcraft, *The song of Hiawatha*. TY Crowell & company, 1899.