A Comparison of Data-Driven Automatic Syllabification Methods

Connie R. Adsett[1,2,⋆] and Yannick Marchand[1,2,⋆⋆]

[1] Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada B3H 1W5
`adsett@cs.dal.ca`

[2] Institute for Biodiagnostics (Atlantic)
National Research Council Canada
1796 Summer Street, Suite 3900
Halifax, Nova Scotia, Canada B3H 3A7
`yannick.marchand@nrc-cnrc.gc.ca`

Abstract. Although automatic syllabification is an important component in several natural language tasks, little has been done to compare the results of data-driven methods on a wide range of languages. This article compares the results of five data-driven syllabification algorithms (Hidden Markov Support Vector Machines, IB1, Liang's algorithm, the Look Up Procedure, and Syllabification by Analogy) on nine European languages in order to determine which algorithm performs best over all. Findings show that all algorithms achieve a mean word accuracy across all lexicons of over 90%. However, Syllabification by Analogy performs better than the other algorithms tested with a mean word accuracy of 96.84% (standard deviation of 2.93) whereas Liang's algorithm, the standard for hyphenation (used in TEX), produces the second best results with a mean of 95.67% (standard deviation of 5.70).

Keywords: Natural language processing, machine learning, automatic syllabification.

1 Introduction

The capability to automatically determine the syllable boundaries in a word is useful for such applications as grapheme-to-phoneme (G2P) conversion and text-to-speech synthesis [1,2]. Data-driven methods are an attractive option to perform automatic syllabification because they are not language specific (unlike syllabification rules), simply requiring a lexicon of syllabified words for training. However, the best method for this task is undetermined.

⋆ The first author was funded by the National Sciences and Engineering Research Council of Canada (NSERC), the National Research Council (NRC) Graduate Student Supplement Program (GSSSP), an Izaak Walton Killam Predoctoral Scholarship, and the Eliza Ritchie Doctoral Scholarship for Women.

⋆⋆ The second author was funded by a discovery grant from NSERC.

© Springer-Verlag Berlin Heidelberg 2009

At best, recent work has only compared algorithms using one language [3,4]. This work compares data-driven automatic syllabification algorithms across nine European languages (Basque, Dutch, English, French, Frisian, German, Italian, Norwegian, and Spanish) in both the spelling and pronunciation domains.

2 Algorithms Compared

The problem of syllabification can be viewed as the task of determining whether or not a syllable boundary exists between each pair of contiguous symbols (letters or phonemes) in a word. A syllable boundary either exists between two symbols (a juncture) or it does not. The syllabification of the entire word can be understood as a structured classification problem because it is formed from the compilation of all juncture classifications. Therefore, the classification of individual junctures in a word are not independent. This is not taken into account in the IB1 and Look-up Procedure algorithms. In contrast, Liang's algorithm, Hidden Markov Support Vector Machines and Syllabification by Analogy incorporate structure information into training and testing.

2.1 Hidden Markov Support Vector Machines

This extension of the general structure SVM framework [5] was introduced by Altun, Tsochantaridis and Hofmann [6]. Intended for structured classification problems like syllabification, the Hidden Markov Support Vector Machine (HM-SVM) approach has produced better results than Syllabification by Analogy on English spelling domain words [3]. This work used the SVM^{hmm} 3.10 package[1].

This method uses a Hidden Markov approach by applying a Viterbi-like algorithm to determine the juncture classification, given previous classifications. The weight vector used in the function that discriminates between possible classification sequences is learned using a Support Vector Machine approach. The best classification sequence gives the maximal output for this function [6].

Training and testing require features for each word juncture. These are all possible substrings within a window of five characters on each side of the juncture. Features are binary values indicating the presence or absence of substrings at each window position. This formulation, along with the classification of junctures using numbered NB (non-boundary or boundary) tags, and the use of 0.1[2] and 0.5 for the C and ϵ parameters, respectively, were based on Bartlett's results [3].

2.2 IB1

When applied to syllabification, the IB1 algorithm (from the instance-based learning algorithm family) compares letter N-grams to determine the appropriate syllabification for each juncture. During training, an N-gram is stored for

[1] This is available from `www.cs.cornell.edu/People/tj/svm_light/svm_hmm.html` (last accessed 9 June, 2009).

[2] For SVM^{hmm} 3.10, the C parameter must be multiplied by the training set size.

each juncture in a word. An N-gram for each juncture in the testing word is then compared to those stored during training to infer the syllabification of the juncture (in the testing word) from the most similar.

A distance measure is used to determine how closely two N-grams match [7,8]. Three feature weighting functions: information gain (IB1-IG), gain ratio (IB1-GR)[3], and chi-squared (IB1-χ^2) can be used to set the values of the weights required for the distance function [8]. Each of these approaches assigns feature weights after training is complete. The values of features in the training data are used to compute the relevance of each feature to classification.

A variety of N-gram sizes were tested to determine which was best. From two characters to the left and right of the juncture (a 4-gram), the size was increased by two on each side up to a 20-gram (the limit of the TiMBL implementation[4]), keeping the number of characters equal on both sides of the juncture.

2.3 Liang's Hyphenation Algorithm

Since Liang formulated his TEX hyphenation algorithm, it has been a standard in the field [9]. Like syllabification, hyphenation is the segmentation of words into substrings.

During training, the `patgen` [9] program is used to generate a set of patterns for the hyphenation of new words[5]. These patterns differ from those used for IB1 and the Look Up Procedure; their length is not restricted and one pattern may contain information for multiple junctures. Once created, they are applied to the words to be syllabified [10].

The algorithm uses many parameters to create patterns, making it impossible to test all parameter settings. Specifically, these parameters are the number of training iterations (between 1 and 9) and, for each iteration, the minimum and maximum substring lengths of the generated patterns (from 1 to 15 characters) along with three values (`good`, `bad`, and `threshold`) used to determine desirable patterns (these may all range from 1 to ∞, in theory). According to Antoš [10], how to tune these parameters is an open problem.

Therefore, this work used parameters selected in previous studies on English hyphenation [9, Table 5], Czech hyphenation [11, Tables 4–9], German compound word hyphenation [12, Tables 4 and 5], and Thai segmentation [10, Table 7], [13, Tables 12.1 and 12.2] for syllabification. A freely available method developed by Ned Batchelder in July of 2007[6] was used to syllabify test words based on the generated patterns. The code was modified slightly to allow for the processing of lists of words using any set of patterns.

[3] This convention is not always followed in the literature and the name IB1-IG is sometimes used to refer to both the Information Gain and the Gain Ratio versions of this algorithm [8].

[4] This is available from `ilk.uvt.nl/timbl/` (last accessed 5 June, 2009).

[5] A version of the program is included by default in Linux distributions.

[6] This is available from `nedbatchelder.com/code/modules/hyphenate.html` (last accessed 9 June, 2009).

2.4 Look Up Procedure

Originally presented as a simpler and superior method to NETtalk for G2P conversion [14], the Look Up Procedure (LUP) has since been applied to automatic syllabification [7]. Except for feature weighting, it operates identically to IB1.

The Look Up Procedure weights are predetermined before testing and the weight set used determines the N-gram size. The 15 sets tested were the same weights used in previous syllabification studies [4] and were originally given by Weijters [14, Figure 2].

2.5 Syllabification by Analogy

Unlike other methods, Syllabification by Analogy (SbA) retains the training set in its entirety. Directed graphs compile the relevant syllabification information obtained lexical entries and are used to syllabify test words. Instances of all test word substrings from the training set are used to create the graph. Graph vertices and edges are labeled with the lexical syllabifications of the substrings (along with the corresponding number of occurences): the substring's initial and final characters form vertex labels and connecting edges are labeled with the intermediate information. The concatenation of the labels of all vertices and edges forming a complete path from the start to the end vertex thus provides a candidate syllabification for the test word.

Only the shortest pathes (without regard to edge weights) are considered and a combination of up to five scoring strategies is used to select the best shortest path. These strategies are the product of the edge weights (these correspond to the substring frequencies in the training data), the standard deviation of values associated with the path structure, the frequency of the same syllabification amongst the shortest paths, the number of differences between one candidate syllabification and all others, and the minimum edge weights. Each scoring strategy may be used independently or with other strategies to rank paths [15]. Because the best of the 31 possible scoring strategies combinations is unknown, all possible combinations were tested.

3 Languages and Lexicons Used

Nine languages were selected due to the availability of lexicons containing marked syllable boundaries. Entries which were clearly non-words, contained non-alphabetic characters or were in any way incomplete were removed. Additionally, proper nouns were removed and homophones and homographs with differing syllabifications to maintain consistency with previous automatic syllabification studies [4]. Because the number of similar entries was a concern, only the infinitive form of verbs and the singular form of nouns were retained in the Italian lexicon. Table 1 summarizes the sources and original and final sizes of the lexicons.

Table 1. The sources and sizes of the lexicons used in this work. S and P denote the domains for which lexicons were used (spelling and pronunciation).

		Number of Entries		
Language	Source	Original	Final	Domain
Basque	EuskalHitzak [16]	100,079	98,913	S,P
Dutch	CELEX [17]	124,136	115,182	S,P
English	CELEX [17]	52,447	31,467	S,P
French	Lexique3 [18]	138,175	31,156	S,P
Frisian	Jelske Dijkstra [19]	63,247	63,219	P
German	CELEX [17]	51,728	20,351	S,P
Italian	Italian Festival [20]	440,084	44,720	S,P
Norwegian	Terje Kristensen [21]	66,992	66,480	S
Spanish	BuscaPalabras [22]	31,491	31,364	S,P

4 Results

Training and testing were performed using 10-fold cross-validation using the same word length distribution in each lexicon fold. Results were computed using word accuracy: the percentage of words automatically syllabified according to the lexicon. After selecting the best parameter setting for each algorithm using the mean word accuracies over all lexicons, the results of each of the five algorithms were compared.

The IB1 algorithm obtained over 80% word accuracy on most lexicons. The highest average word accuracy was given by IB1_IG and the three largest N-gram sizes gave similar results with the best performance (94.36%) achieved using 12-grams.

The mean word accuracies from Liang's algorithm for each parameter set span a wide range of values; from 2.45% to 95.48% . Overall, the parameter settings previously used for Czech hyphenation [11, Table 9] give the best average word accuracy (95.48%).

Like the results of Liang's algorithm, those given by the Look Up Procedure range from 11.89% to 91.44%. Best average word accuracy across lexicons is achieved using the weights [1, 4, 16, 64, 16, 5, 1] which occur most in the top three for word accuracy for each lexicon (14 times). Average word accuracy using these weights is 91.44%.

In the case of HM-SVM, only one parameter setting was tested. These results give an average word accuracy of 95.17%.

All 31 scoring strategy combinations were tested for SbA. The best (using the frequency of the same syllabification and the largest minimum edge weight) achieved a mean word accuracy of 96.70%. At the individual lexicon level, this same combination again rose above the others with 11 occurrences in the top three scores for each lexicon.

Of the 16 lexicons tested, Liang's algorithm gave the best results for eight, and both IB1 and SbA performed best for four. Interestingly, although the IB1 algorithm treats each juncture classification as independent, it still produces the

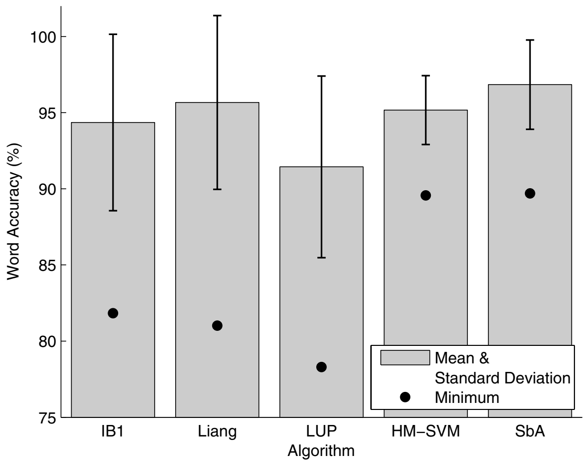

Fig. 1. Comparison of the minimum, mean and standard deviation word accuracy results of each algorithm

most accurate results for some lexicons. In fact, for nine of the 16 lexicons, the IB1 results are better than at least one of the algorithms that included structure information in the classification process. This difference in the algorithms does not seem to be the main key to accurate automatic syllabification.

These results point to Liang's algorithm as the best choice for automatic syllabification. However, for three lexicons, this approach provides the worst syllabifications. The poor performance of this algorithm on the French spelling domain (81.02%) and Spanish pronunciation domain (83.26%) point to why this algorithm does not give the overall best mean word accuracy (Figure 1).

Figure 1 also shows the minimum word accuracies and standard deviations of the mean word accuracies of the algorithms. Except for Liang's algorithm, which obtained minimum word accuracy in the French spelling domain, the English spelling domain lexicon was the source of the minimum word accuracies. This is not surprising, given that English syllabification is thought to be more complex (allowing a wider variety of syllabic structures) than other languages. HM-SVM and SbA have lower standard deviations and much higher minimum word accuracies than the other three algorithms because these methods are better able to model the syllabification task regardless of the differing language characteristics. Of the two, SbA gives the highest mean word accuracy which is also significantly better than Liang's algorithm ($\chi^2_{obt} = 1735.17$, $p < 0.0001$).

These results differ from previous findings which showed that the HM-SVM method outperformed SbA for automatic syllabification in the English spelling domain [3]. This may be because training and test methods used were different (prior work used a training set of 14,000 words and a test set of 25,000 words from CELEX [17]).

Overall, these results point to SbA as a better choice for a syllabification algorithm.

5 Conclusions and Future Work

This work has shown, using a wide range of languages in both the spelling and pronunciation domains, that some data-driven algorithms automatically syllabify words better regardless of the lexicon characteristics. All algorithms achieve mean word accuracies over 90% but, from these results, the Syllabification by Analogy approach appears to be the best choice for automatic syllabification tasks. Although not producing as high word accuracy, Hidden Markov Support Vector Machines have also been shown to be more consistent in syllabification results. Liang's algorithm, although performing well for some lexicons, also syllabifies much less accurately for some lexicons. This same variability in results across lexicons is seen in the IB1 and Look-up Procedure algorithms.

Deeper investigation into how optimal parameters can be chosen is especially necessary for Liang's algorithm and the Look Up Procedure, given the large feature spaces of these two methods. Furthermore, by nature, data-driven algorithms require a lexicon of syllabified words in order to be used but the impact of lexicon size is unknown. Testing increasing lexicon sizes in each language and for each algorithm would provide insight into which algorithm is capable of learning syllable boundaries, given as little data as possible. Finally, it has been demonstrated that automatic syllabification information improves English grapheme-to-phoneme conversion accuracy [1]. Testing this with additional languages would clarify whether this finding generalizes to other languages.

References

1. Bartlett, S., Kondrak, G., Cherry, C.: Automatic syllabification with structured SVMs for letter-to-phoneme conversion. In: Proceedings of ACL 2008: HLT, Columbus, Ohio, pp. 568–576 (2008)
2. Libossek, M., Schiel, F.: Syllable-based text-to-phoneme conversion for German. In: Proceedings of the Sixth International Conference on Spoken Language Processing (ICSLP 2000), Beijing, China, pp. 283–286 (2000)
3. Bartlett, S.E.: A discriminative approach to automatic syllabification. Master's thesis, Department of Computing Science, University of Alberta (2007)
4. Marchand, Y., Adsett, C.R., Damper, R.I.: Automatic syllabification in English: A comparison of different algorithms. Language and Speech 52(1), 1–27 (2009)
5. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: Proceedings of the 21st International Conference on Machine Learning (ICML 2004), Banff, Canada, pp. 104–112 (2004)
6. Altun, Y., Tsochantaridis, I., Hofmann, T.: Hidden markov support vector machines. In: Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003), Washington, DC, pp. 3–10 (2003)
7. Daelemans, W., van den Bosch, A.: Generalization performance of backpropagation learning on a syllabification task. In: Drossaers, M.F.J., Nijholt, A. (eds.) TWLT3: Connectionism and Natural Language Processing, Enschede, The Netherlands, pp. 27–37 (1992)
8. Daelemans, W., Zavrel, J., van der Sloot, K., van den Bosch, A.: TiMBL: Tilburg Memory-Based Learner, 6.0th edn., Tilburg, The Netherlands (2007)

9. Liang, F.M.: Word Hy-phen-a-tion by Com-put-er. PhD thesis, Stanford University, Palo Alto, CA (1983)
10. Antoš, D.: PatLib, pattern manipulation library. Master's thesis, Faculty of Informatics, Masaryk University Brno (2001)
11. Sojka, P., Ševeček, P.: Hyphenation in TEX - quo vadis? TUGboat 16(3), 280–289 (1995)
12. Sojka, P.: Notes on compound word hyphenation in TEX. TUGboat 16(3), 290–297 (1995)
13. Sojka, P., Antoš, D.: Context sensitive pattern based segmentation: A Thai challenge. In: Proceedings of EACL 2003 workshop Computational Linguistics for South Asian Languages – Expanding Synergies with Europe, Budapest, Hungary, April 2003, pp. 65–72 (2003)
14. Weijters, A.J.M.M.: A simple look-up procedure superior to NETtalk? In: Proceedings of the International Conference on Artificial Neural Networks (ICANN 1991), Espoo, Finland, pp. 1645–1648 (1991)
15. Marchand, Y., Damper, R.I.: A multistrategy approach to improving pronunciation by analogy. Computational Linguistics 26(2), 195–219 (2000)
16. Perea, M., Urkia, M., Davis, C.J., Agirre, A., Laseka, E., Carreiras, M.: E-Hitz: A word frequency list and a program for deriving psycholinguistic statistics in an agglutinative language (Basque). Behavior Research Methods 38(4), 610–615 (2006)
17. Baayen, R.H., Piepenbrock, R., Gulikers, L.: The CELEX lexical database (CD-ROM). Technical report, Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA (1995)
18. New, B., Pallier, C.: Manuel de Lexique 3, France. 3.03 edn. (2005)
19. Dijkstra, J., Pols, L.C.W., Van Son, R.J.J.: Frisian TTS, an example of bootstrapping TTS for minority languages. In: Proceedings of the 5th ISCA Speech Synthesis Workshop, Pittsburgh, pp. 97–102 (2004)
20. Cosi, P., Tesser, F., Gretter, R., Avesani, C.: Festival speaks Italian? In: Proceedings of Eurospeech 2001, Aalborg, Denmark, pp. 509–512 (2001)
21. Kristensen, T.: A neural network approach to hyphenating Norwegian. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000), Como, Italy, vol. 2, pp. 148–153. IEEE, Los Alamitos (2000)
22. Davis, C.J., Perea, M.: BuscaPalabras: A program for deriving orthographic and phonological neighborhood statistics and other psycholinguistic indices in Spanish. Behavior Research Methods 37(4), 665–671 (2005)