Data-Driven Approaches for Automatic Detection of Syllable Boundaries

Jilei Tian

Audio-Visual Systems Laboratory
Nokia Research Center, Tampere, Finland
jilei.tian@nokia.com

Abstract

Syllabification is an essential component of many speech and language processing systems. The development of automatic speech recognizers frequently requires working with subword units such as syllables. More importantly, syllabification is an inevitable part of speech synthesis system. In this paper we present data-driven approaches to supervised learning and automatic detection of syllable boundaries. The generalization capability of the learning is investigated on the assignment of syllable boundaries to phoneme sequence representation in English. A rule-based self-correction algorithm is also proposed to automatically correct some syllabification errors. We conducted a series of experiments and the neural network approach is clearly better in terms of generalization performance and complexity.

1 Introduction

The development of speech synthesizers and speech recognizers often requires working with subword units such as syllables. For instance, robust speech recognition often makes use of word spotters based on syllables for detecting out-of-vocabulary speech and for modeling unknown words in spontaneous speech. Syllabification plays a key role in any text-to-speech (TTS) system [1]. In many languages the pronunciation of phonemes is a function of their location in the syllable relative to the syllable boundaries. Location in the syllable also has a strong effect on the duration of the phone, and is therefore a crucial piece of information for any model of segmental duration. Syllable is also used to assign the strong/weak stress on the word, e.g. the English noun Insult vs. the related verb inSULT.

The system being developed requires study on the algorithm for dividing words into syllables, due to the fact that a commonly accepted algorithm shared among the linguistic community does not exist. In the linguistic literature, we can find grammatical rules or attempts to explain the division of words into syllables [2]; it usually requires comprehensive linguistic knowledge. Probabilistic context-free grammar (PCFG) was proposed in [3], but the performance seems having a room for improvement. It also requires a larger training database to get statistical information. The neural network applied to the assignment of syllable boundaries to orthographic representation was also presented in [4]. Precisely it hyphens the word, rather than syllabification in the phoneme sequence. Since the phoneme sequence, rather than orthographic representation, contains more information to extract spoken syllables, the reported performance is limited. In recent years finite-state transducers (FSTs) have become increasingly popular as a flexible and mathematically elegant computational model for the conversion and mapping between symbol strings, most notably in the domains of phonology and morphology [1]. Paper [5] presents a finite-state model for syllabification. The syllabifier is implemented as a weighted finite-state transducer. The transducer is constructed by obtaining syllables as well as their structure and frequencies from a training database. Observed frequencies of onset, nucleus and coda types are converted into weights, which are then associated with the pertinent transitions between states in the transducer. It requires a large training database and generalization capability for unseen data is not studied.

In this paper, we investigate data-driven approaches for modeling syllabification. Decision tree- and neural network-based approaches are compared and evaluated. The study is mainly focused on the generalization capability of the learning. It is aiming to train a robust model that can be used for various types of data. Then it is feasible in practice to label small amount of training data without profound linguistic knowledge. With the help of general rules, the self-correction algorithm is introduced. Our results indicate that a suitable neural network provides clearly better overall performance and smaller model size than decision trees. Particularly, the generalization performance of neural networks is superior.

The remainder of the paper is organized as follows. We first describe how to structure syllables in Section 2. After that, we briefly review both the decision tree- and the neural network-based methods. The self-correction algorithm is also introduced in Section 3. The performance of the proposed approaches is evaluated in terms of generalization capability, complexity and syllabification accuracy in Section 4, with concluding remarks in Section 5.

2 Syllable structure

A syllable is a basic unit of word studied on both the phonetic and phonological levels of analysis. It is typically composed of more than one phoneme. No matter how easy it can be for people and even for children to count the number of syllables in a sequence in their native language, still there are no universally agreed upon phonetic definitions of what a syllable is. It is phonologically believed that syllable is a complex unit made up of nuclear and marginal elements. Nuclear elements are the vowels or syllabic segments, and marginal elements are the consonants or non-syllabic segments. Standard dictionaries provide syllabification that is influenced by the morphological structure of words; it is common in such dictionaries to split prefixes and suffixes from stems. For instance, some dictionaries syllabify the word *glamour* as [g l ae m – er], whereas the more plausible syllabification in speech is [g l ae -

m er]. Our output produces the latter and hence is more faithful to spoken syllables. Syllabification ought to represent the properties of spoken utterances, rather than morphological structure, particularly true for ASR and TTS systems.

A syllable can be described by a series of grammars [3]. The simplest grammar is the phoneme grammar, where a syllable is tagged with the corresponding phoneme sequence. The consonant-vowel grammar describes a syllable as a consonant-vowel-consonant (CVC) sequence. The syllable structure grammar divides a syllable into onset, nucleus and coda (ONC) as shown in *Figure 1*. The nucleus is obligatory which can be either a vowel or a diphtong. An onset is the first part of a syllable consisting of consonants and ending to the nucleus of the syllable. e.g. /t/ is the onset of the syllable [t eh k s t]. A nucleus is the vowel part of a syllable, e.g. /eh/ in the syllable. A coda is the part of a syllable that follows the nucleus. A coda is constructed of consonants, e.g. /k s t/ in the syllable. The nucleus and coda are combined to form the rhyme of a syllable. A syllable has a rhyme, even if it doesn't have a coda.

In the syllable structure grammar, the consonants are assigned as onset or coda. It contains more information than the CVC structure for multi-syllable words. The syllable structure grammar is used throughout the paper. The phoneme sequence is mapped into its ONC representation. The model is trained on the mapping. In the decoding phase, given a phoneme sequence, the ONC sequence is first generated, and then the syllable boundaries are uniquely decided on the ONC sequence. For invalid ONC sequences, a self-correction algorithm is applied to solve the problem by utilizing certain common linguistic rules. Now, the syllabification task is divided into the following steps.

1. The pronunciation phoneme string is mapped to ONC string, for example:

(word) text -> (pronunciation) t eh k s t -> (ONC) O N C C C

2. Training on the data in the format of "pronunciation -> ONC"
3. Given the pronunciation, the corresponding ONC sequence is generated from the model. Then syllable boundaries are placed at the location starting with symbol "O" or "N" if it is not preceded with symbol "O".

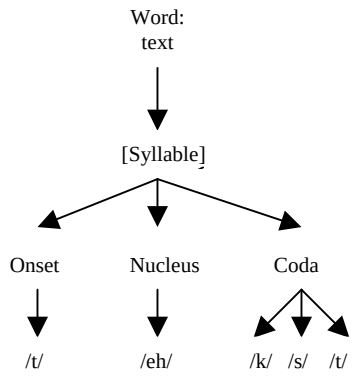

Figure 1. Diagram of the syllable structure grammar.

3 Data-driven syllabification

3.1 Neural network-based syllabification

The basic neural network-based ONC model is a standard multi-layer perceptron (MLP), as seen in *Figure 2*. Phonemes are presented to the MLP network one at a time in a sequential manner. The network gives estimates of ONC posterior probabilities for each presented phoneme. In order to take the phoneme context into account, a number of phonemes on each side of the phoneme in question are also used as inputs to the network. Thus, a window of phonemes is presented to the neural network as input. *Figure 2* shows a typical MLP with a context size of w phonemes $ph_{i-w}...ph_{i+w}$ centered at phoneme ph_i. The centermost phoneme ph_i is the phoneme that corresponds to the output of the network. Therefore the output of the MLP is the estimated ONC probability $P(onc_k|ph_{i-w},...,ph_{i+w})$ ($onc_k \in \{O, N, C\}$) for the centermost phoneme ph_i in the given context $p_{i-w}...p_{i+w}$. A phonemic null is defined in the phoneme set and is used for representing phonemes to the left of the first phoneme and to the right of the last phoneme in a pronunciation.

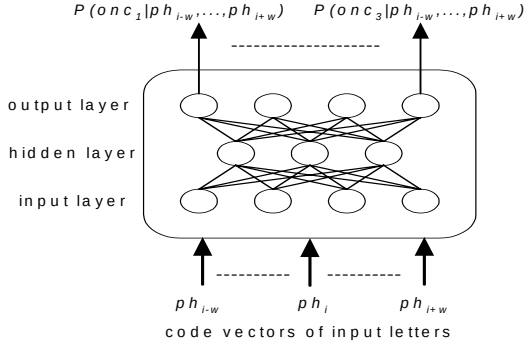

Figure 2. Two-layer neural network architecture.

The ONC neural network is a fully connected MLP, which uses a hyperbolic tangent sigmoid shaped function in the hidden layer and a softmax normalization function in the output layer. The softmax normalization ensures that the network outputs are in the range [0,1] and sum up to unity.

$$P_i = \frac{e^{y_i}}{\sum_{j=1}^{3} e^{y_j}}. \quad (1)$$

In Equation (1), y_i and P_i denote the i^{th} output value before and after softmax normalization. It has been shown in [6] that a neural network with softmax normalization will approximate class posterior probabilities when trained for one-out-of-N classification and when the network is sufficiently complex and trained to a global minimum. Since the neural network input units are continuous valued, the phonemes in the input window need to be transformed to some numeric quantity. An example of an orthogonal codebook representing an alphabet used for ONC mapping task is shown in *Table 1*. The last row in the table is the code for the phonemic null. The orthogonal code has an equal size to the number of phonemes in the alphabet. An important

property of the orthogonal coding scheme is that it does not introduce any correlation between different letters.

The ONC neural network is trained by the standard back-propagation (BP) algorithm augmented by a momentum term. Each phoneme with context and the corresponding ONC tag of the pronunciation make up one training example. Weights are updated in a stochastic on-line fashion. Before testing the models, all parameters are rounded off to eight bits as this was found sufficient for representing model parameters without a significant loss in accuracy. The number of parameters in the models therefore equals the required memory for storage in bytes.

Table 1. Orthogonal phoneme coding scheme.

Letter	Code
aa	100...0000
ae	010...0000
...	...
B	000...1000
P	000...0100
T	000...0010
#	000...0001

The outputs of the ONC neural network approximate the ONC posterior probabilities corresponding to the centermost phoneme. The ONC sequence of a pronunciation is obtained by combining the network outputs for each individual phoneme in the pronunciation. Given a pronunciation with phonemic representation, the ONC tag of phoneme ph_i is given by

$$onc = \underset{onc_k}{\operatorname{argmax}}\{P(onc_k \mid ph_{i-w},..., ph_{i+w})\}, \quad (2)$$

where $P(onc_k \mid ph_{i-w},..., ph_{i+w})$ is the network output corresponding to onc_k given input phonemes $ph_{i-w}...ph_{i+w}$, and variable w denotes the phoneme window context size, respectively. Variable *onc* takes value from set of [*O N C*].

3.2 Decision tree-based syllabification

Contrary to the neural network approach, a separate decision tree is trained for each of the different phonemes. The ONC tag for a phoneme is obtained by "asking" a series of questions about the context of the phoneme in question as defined by the corresponding decision tree. A decision tree is composed of a root node, internal nodes and leaves. In the trees used here, the context is defined by the neighboring phonemes. Each node contains information about the attribute and ONC identity.

In the decoding phase, a ONC tag sequence is generated by going through the pronunciation phoneme by phoneme from left to right. The decision tree corresponding to the current phoneme is climbed based on the context information until a leaf is reached. The ONC tag that corresponds to the current phoneme is read from the leaf. Then the process moves on to the next phoneme and the ONC tag for this phoneme is found in a similar way.

When a decision tree is trained for a given phoneme, all the training cases for the phoneme are considered. A training case for the phoneme is composed of the phoneme context and the corresponding ONC tag of the pronunciation. During training, the decision tree is grown and the nodes of the decision tree are split into child nodes according to an information theoretic optimization criterion. Details about decision tree training can be found in [7][8].

3.3 Self-correction algorithm

Syllabification from an ONC sequence can be obtained by writing a very simple declarative grammar to decode the locations of syllable boundaries in polysyllabic pronunciations. The grammar describes that each pronunciation consists of one or more syllables of the structure ONC, i.e., of an obligatory syllable nucleus (N) optionally preceded or followed, or both, by any number of consonants (O or C). The self-correction algorithm is triggered by finding invalid pattern /O C/ from the ONC string. If found, certain linguistic rules are applied, such as,

(1). By assigning a higher cost to the last consonant of each syllable, we can enforce the syllable boundary to be placed as early as possible, thereby implementing the well-known maximal onset principle.

(2). It is also assumed that error in the ONC sequence is minimized.

(3). In English, the onset can contains up to three phonemes and coda up to four. This rule can also be implemented into the algorithm.

The more detailed linguistic rules we have, the better performance we can expect from the algorithm. The self-correction algorithm is summarized in *Figure 3*.

1. Read ONC string mapped from a pronunciation
2. if ONC string matches invalid pattern /O C/, goto 3
   else goto 4
3 Applying rules 1, 2, 3 to decide mapping eigher /O C/->/C C/ or /O C/->/O O/, goto 4,
4. if corrected ONC string matches invalid pattern, goto 3, else output the corrected ONC string.

Figure 3. The self-correction algorithm.

4 Experimental results

4.1 Setup

The neural network- and decision tree-based syllabification methods are evaluated on the CMU dictionary for US English. The dictionary contains 108,080 words with pronunciation and labels with ONC information. The pronunciation and mapped ONC sequence part of the dictionary are extracted to form the training data. The whole dictionary is further arbitrarily split into independent training and test sets without overlapping. The dictionary contains generic words as well as a few proper names. With preliminary experiments a suitable context length of 2 phonemes was chosen. Therefore both the neural network and decision tree methods use a context of 2 phonemes to the left and right of the centermost phoneme.

4.2 Generalization capability of NN

In practice, only a small amount of training data can be easily obtained by manual annotation. Since the data-driven approaches require such training data, so it would be very beneficial if model trained on the small training data can be

used on the large database with acceptable loss of accuracy. Figure 4 shows the correct string rate of neural network-based pronunciation-to-ONC mapping on the test set with different training set. The neural network uses five hidden units and three output units. The number of inputs is 200 corresponding to a 5-phoneme input window and 40 different phonemes (including the phonemic null). With 8-bit precision for the network weights this corresponds to a memory requirement of about 1,035 Bytes. It can be seen that neural network method leads to a quick saturation in performance as a amount of train data increases. Very good performance on the test set can be obtained with a small amount of train data, e.g. about 2000 samples leading to >99% string rate. For the training data, the string rates are 100% for all tested cases.

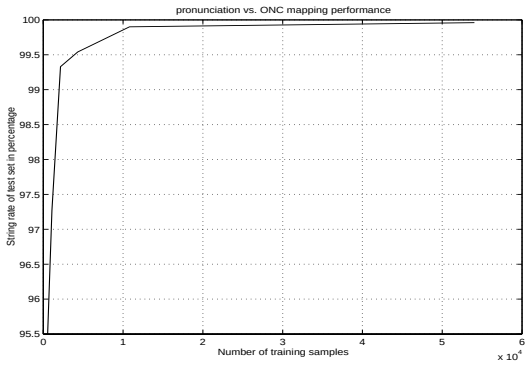

Figure 4. Neural network performance of pronunciation vs. ONC sequence mapping.

4.3 Comparison between NN and DT

The generalization capability is first compared between the NN- and DT-based methods. As show in Section 4.2, the performance of NN-based method is saturated with a training set of about 2000 samples. Thus training process has taken 2000 samples into use for both NN and DT models.

Table 2. Performance comparison between NN and DT.

	String Rate: training set	String Rate: Test set	Model Size
NN	100%	99.33%	1035 Bytes
DT	100%	97.86%	4407 Bytes

Table 2 clearly indicates that NN method outperforms DT method in all comparison including model size and generalization capabilities. It should be noted that DT model size is obtained by using optimization mentioned in [8].

4.4 Evaluation of self-correction algorithm

The self-correction algorithm is developed to automatically correct the ONC sequence containing invalid pattern to make detection of syllable boundaries possible. The NN models are trained on the data with different number of samples. The experiments are carried out on the remaining test set. Among the invalid errors, *Figure 5* shows the number of errors that have been corrected. The dashed line stands for the perfect correction and solid line indicates the excellent performance of the self-correction algorithm. The corrected rate (equivalent to the slope of fitted solid line) is 89.1% in average. 20% of total errors are invalid errors, so the algorithm improves about 20% of the syllabification rate.

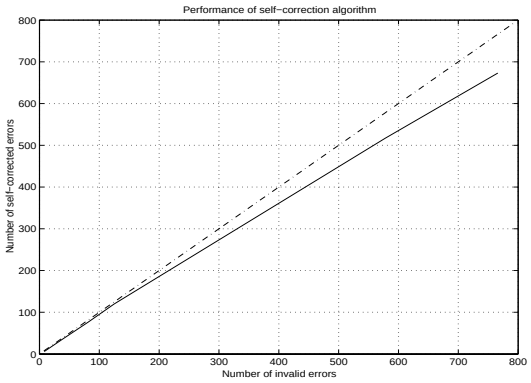

Figure 5. Performance of self-correction algorithm.

5 Conclusions

The syllabification is an important problem in automatic speech recognition and speech synthesis applications. In this paper we have compared NN- and DT-based methods. Our results show that NN provides clearly better overall performance and smaller model size compared to DT. Especially the generalization performance of NN is superior to that of the DT. A self-correction algorithm has also been presented and proven to be useful in eliminate the effects of linguistically invalid ONC patterns.

6 References

[1] Sproat, R., *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*. Kluwer, Dordrecht, 1998.

[2] Kahn, D., *Syllable-Based Generalizations in English Phonology*, Doctoral Dissertation, Massachusetts Institute of Technology, USA, 1976.

[3] Müller, K., "Automatic Detection of Syllable Boundaries Combining the Advantages of Treebank and Bracketed Corpora Training", in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France, 2001.

[4] Daelemans, W. and van den Bosch, A. "Generalization performance of backpropagation learning on a syllabification task", in *Proceedings of Twente Workshop on Language Technology 3: Connectionism and Natural Language Processing*, Twente, The Netherlands, 1992.

[5] Kiraz, G. A., and Möbius, B., "Multilingual Syllabification Using Weighted Finite-State Transducers", in *Proceedings of the 3rd ISCA Speech Synthesis Workshop*, Australia, 1998.

[6] Bishop, C., *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, UK, 1995.

[7] Quinlan, J., *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., San Mateo, CA, 1993.

[8] Suontausta, J. and Tian, J., "Low memory decision tree method for text-to-phoneme mapping", in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, U.S. Virgin Islands, 2003.