OPTIMAL SUBSET SELECTION FROM TEXT DATABASES

Jilei Tian, Jani Nurminen and Imre Kiss

Multimedia Technologies Laboratory
Nokia Research Center, Tampere, Finland
{jilei.tian, jani.k.nurminen, imre.kiss}@nokia.com

ABSTRACT

Speech and language processing techniques, such as automatic speech recognition (ASR), text-to-speech (TTS) synthesis, language understanding and translation, will play a key role in tomorrow's user interfaces. Many of these techniques employ models that must be trained using text data. In this paper, we introduce a novel method for training set selection from text databases. The quality of the training subset is ensured using an objective function that effectively describes the coverage achieved with the strings in the subset. The validity of the subset selection technique is verified in an automatic syllabification task. The results clearly indicate that the proposed systematic selection approach maximizes the quality of the training set, which in turn improves the quality of the trained model. The presented idea can be used in a wide variety of language processing applications that require training with text databases.

1 INTRODUCTION

Most automatic speech recognition (ASR) and text-to-speech (TTS) systems contain models that have to be trained with text data. Typical examples can be found from many parts of the systems. In pronunciation modeling, some data-driven approach, such as neural network based methods or decision tree based methods [6], are often applied, especially for languages like English. These statistical models are trained using a pronunciation dictionary containing grapheme-to-phoneme entries. In text-based language identification [8], the model is trained using a multilingual text corpus that consists of word entries from the target languages. In the data-driven syllabification task [7], the model is trained using text-based pronunciations and the corresponding syllable structures.

In all data-driven approaches, the selection of a suitable training set can be regarded as a very important step in the training process. In general, the performance of any trained model depends quite strongly on the quality of the text data used in the training. With text-based data, the importance of the training set selection is very pronounced since the generation of the training data entries is often very time and resource consuming and requires language-specific skills. In this paper, we show that systematic training set selection results in enhanced model performance and/or offers the possibility to use a smaller training set size. In practice, the reduced training set size brings two significant additional benefits. First, the amount of manual annotation work is reduced, which in turn decreases the probability of errors and inconsistencies in the annotations. Second, the memory consumption and the computational load caused by the training process are lowered. In some cases this advantage propagates to the trained model as well; the size of a decision tree model, for example, depends on the size of the training set.

Despite the evident importance of the training set selection, this step is often neglected in practice. Usually, the training set is obtained by collecting a set of random entries from a larger text database or by decimating a sorted corpus. The drawback of these solutions is that the amount of meaningful information in the selected text data set is not maximized. The random selection method is rather coarse and does not produce consistent results. The method of decimating a sorted data corpus, on the other hand, only uses a limited number of the initial characters of the strings and thus does not guarantee good performance.

In this paper, we present a method that can quasi-optimally select a subset from a text database in such a manner that the text coverage is maximized. To achieve this, we define an objective function that is optimized in the subset selection. The objective function measures the "subset distance" using the generalized Levenshtein distances between the text strings. This paper also introduces an algorithm for optimizing the objective function. For practical applications with large databases, the algorithm can be modified in order to speed up the processing or to lower the memory consumption, but the main idea and the objective function will remain useful in all cases. To demonstrate the usefulness of the proposed approach, we evaluate it in the syllabification task.

The text subset selection method introduced in this paper can be used in a wide variety of different applications. One good example is the language identification task [8], in which the proposed approach makes it possible to easily balance the number of training set entries from each target language while at the same time giving a good coverage for every target language. In addition to the training set selection task discussed extensively in this paper, it is possible to employ the same techniques for clustering a text database. Moreover, when used together with a meaningful distance measure, such as the generalized Levenshtein distance, the proposed approach enables the use of vector quantization techniques on text data.

The remainder of the paper is organized as follows. We first describe the generalized Levenshtein distance and introduce the basic principles of the text database selection algorithm in Section 2. In Section 3, we describe the syllabification task used as the practical example by briefly reviewing the syllable structure grammar and the neural network based syllabification method. The performance of the proposed subset selection approach is evaluated in the

syllabification task in Section 4. Finally, some concluding remarks are presented in Section 5.

2 SELECTION ALGORITHM

In order to be able to select a subset from a text database in a systematic and meaningful manner, an objective function measuring the quality of the subset must be defined. The objective function should somehow measure the similarity or the dissimilarity of the entries. In the proposed approach, we base the objective function on the generalized Levenshtein distance. In this section, we first describe the basic properties of this distance measure and then continue by defining an objective function measuring the average distance within a subset and by introducing an algorithm for selecting subsets of different sizes in a quasi-optimal manner.

2.1 Generalized Levenshtein distance

The generalized Levenshtein distance (GLD) is defined as the minimum cost of transforming one string into another by means of a sequence of basic transformations: insertion, deletion and substitution [4]. The transformation cost is determined by the costs assigned to each basic transformation.

Let x and y be strings of length m and n, respectively, whose symbols belong to a finite alphabet of size s. Let x_i be the ith symbol of string x, with $1 \le i \le m$, and $x(i)$ be the prefix of the string x of length i, i.e. the substring containing the first i symbols of x. In addition, let $d(i,j)$ be the distance between $x(i)$ and $y(j)$, and ε be an empty string. Furthermore, we denote by $w(a,b)$, $w(a,\varepsilon)$ and $w(\varepsilon,b)$ the cost of substituting the symbol a with the symbol b, the cost of deleting a and the cost of inserting b, respectively. The distance $d(m,n)$ is recursively computed based on the definitions of $d(0,0)$, $d(i,0)$ and $d(0,j)$ ($i = 1\ldots m$, $j = 1\ldots n$), representing the initial distance, the cost of deleting the prefix $x(i)$ and the cost of inserting the prefix $y(j)$, respectively, as follows:

$$d(0,0) = 0$$
$$d(i,0) = d(i-1,0) + w(x_i, \varepsilon) \qquad \forall i = 1\ldots,m \tag{1}$$
$$d(0,j) = d(0,j-1) + w(\varepsilon, y_j) \qquad \forall j = 1\ldots,n$$

$$d(i,j) = \min\begin{cases} d(i-1,j) + w(x_i, \varepsilon) \\ d(i,j-1) + w(\varepsilon, y_j) \\ d(i-1,j-1) + w(x_i, y_j) \end{cases} \tag{2}$$

The original Levenshtein distance is characterized by the following costs: $w(a, \varepsilon) = 1$, $w(\varepsilon,b) = 1$, and $w(a, b)$ is 0 if a is equal to b and 1 otherwise. Its generalized version assumes that different costs can be associated to transformations involving different symbols. In the case of an alphabet of s symbols, this requires a table of size (s+1) times (s+1), called the cost table, to store all the substitution, insertion and deletion costs. It can be shown that the defined distance is a metric if the cost table is symmetric.

2.2 Objective function and selection algorithm

In our approach, we measure the quality of a text subset using an objective function based on the generalized Levenshtein distance. As described in Section 2.1, the Levenshtein distance can be used for measuring the distance between any pair of entries. Similarly, the distance for the whole text data set can be calculated by averaging the distances of all the string pairs in the set. Suppose that there are m entries in the database and the ith entry is denoted by $e(i)$. With these definitions, we can compute the overall "subset distance" D as:

$$D = \frac{2 \cdot \sum_{i=1}^{m} \sum_{j>i}^{m} ld(e(i), e(j))}{m \cdot (m-1)}, \tag{3}$$

where $ld(e(i), e(j))$ is the GLD between the ith and jth entries.

Based on the above objective function, it is possible to design an algorithm that selects a subset from a text database in such a manner that the distance D is maximized. The following algorithm recursively constructs the subset by always selecting the new entry that maximizes the distance to the other selected entries.

1. Calculate the Levenshtein distances for all the pairs; $ld(e(i), e(j))$;
2. Initially select the pair that has the largest distance among all pairs in the database,
$$((subset\_e(1), subset\_e(2)) = \underset{(1 \le i \le m, j > i)}{\operatorname{argmax}} \{ld(e(i), e(j))\}. \tag{4}$$
3. Assuming that the selected subset has k entries (in the first time k = 2), the target now is to find the k+1-th entry to the subset. The selection that approximately maximizes the amount of new information brought into the subset can be done using the following formula.
$$p = \underset{(1 \le i \le m)}{\operatorname{argmax}} \left\{ \sum_{j=1, e(i) \ne subset\_e(j)}^{k} ld(e(i), subset\_e(j)) \right\}. \tag{5}$$
The selected entry p is added into the subset as $subset\_e(k+1)$.
4. Repeat step 3 until the preset subset size is reached.

3 EXAMPLE APPLICATION: SYLLABIFICATION TASK

The development of speech synthesizers and speech recognizers often requires working with sub-word units such as syllables [5]. We have earlier described a neural network based approach for the automatic assignment of syllable boundaries in [7]. In this paper, we revisit the topic and use this syllabification task for verifying the usefulness of the proposed subset selection approach. The first part of this section gives some basic information on the task and the second part discusses the neural network approach. The practical results achieved in this task are presented in Section 4.

3.1 Syllable structure

A syllable is a basic unit of word studied on both the phonetic and phonological levels of analysis [2]. The syllable information can be described using 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, on the other hand, divides a syllable into onset, nucleus and coda (ONC) as shown in Figure 1. The nucleus is an obligatory part that can be either a vowel or a diphthong. The onset is the first part of a syllable consisting of consonants and ending at the nucleus of the syllable, e.g. in the syllable [*t eh k s t*], /*t*/ is the onset and the vowel part /*eh*/ is the nucleus. The part of a syllable that follows the nucleus forms the coda. The coda is constructed of consonants, e.g. /*k s t*/ in

our example 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. The ONC representation used in the syllable structure grammar contains more information than the CVC structure for multi-syllable words. The syllable structure grammar was used in [7] and it is also used in this paper.

In the automatic syllabification task, the phoneme sequences are mapped into their ONC representations. The data-driven syllabification model is trained on the mapping information. 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 [7] can be applied to solve the problem by utilizing certain common linguistic rules. The whole syllabification task can be summarized as follows:

1. Each pronunciation phoneme string in the training set is mapped into the corresponding ONC string, for example:
(word) *text* -> (pronunciation) *t eh k s t* -> (ONC) *O N C C C*
2. The model is trained on the data in the format of "pronunciation -> ONC"
3. Given a pronunciation string, the corresponding ONC sequence is generated using the model. Then, the syllable boundaries are placed at the location starting with symbol "*O*", or with "*N*" if it is not preceded with symbol "*O*".

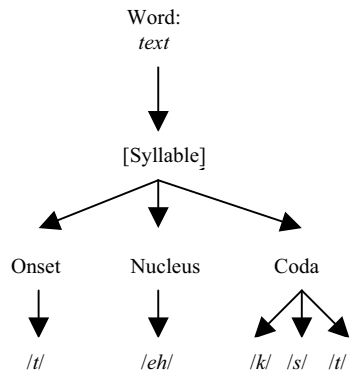

Figure 1. Diagram of the syllable structure grammar.

3.2 Neural network based syllabification approach

The basic neural network based ONC model presented in [7] is a standard multi-layer perceptron (MLP) shown in Figure 2. The input phonemes are presented to the MLP network 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.

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,

$$P_i = \frac{e^{y_i}}{\sum_{j=1}^{3} e^{y_j}}. \quad (6)$$

In Equation (6), y_i and P_i denote the ith output value before and after softmax normalization. It has been shown in [1] 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 text-valued, the phonemes in the input window need to be transformed to some numeric quantity. This can be done, for example, using an orthogonal codebook representing the alphabet used for the ONC mapping task, as shown in Table 1. The last row in the table is the code for the phonemic null. An important property of the orthogonal coding scheme is that it does not introduce any correlation between the different letters.

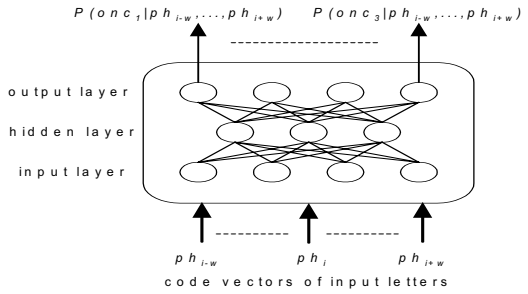

Figure 2. Two-layer neural network architecture.

The ONC neural network is trained using 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. All parameters are rounded off to eight bits as this was found sufficient for representing model parameters.

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 its

phonemic representation, the ONC tag of phoneme ph_i is given by

$$onc = \underset{onc_k}{\operatorname{argmax}}\{P(onc_k \mid ph_{i-w},..., ph_{i+w})\}, \quad (7)$$

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

4 EXPERIMENTAL RESULTS

The neural network based syllabification method is evaluated using the CMU dictionary for US English. The dictionary contains 10,801 words with their pronunciations and labels with ONC information. The pronunciations and the mapped ONC sequences are extracted to form the training data. The training set is selected from the whole database by using the following methods:

- Decimation of the sorted dictionary (denoted as DECIMATE);
- Subset selection from the text database using the selection approach proposed in this paper (denoted as SELECT).

With both methods, the data not selected to the training set constitutes the test set.

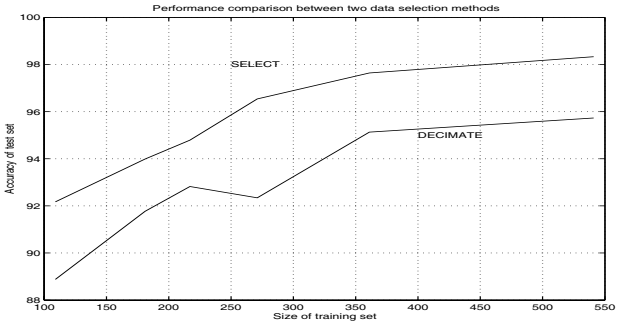

Figure 3. ONC accuracy on test set with different training set sizes using the two data selection methods.

Figure 3 shows the experimental results achieved using the two data selection methods. The efficiency of the training set selection approach can be studied by evaluating the generalization capability. The general rule of thumb is that the more training data is available, the better performance can be expected. However, the selection of the training data affects the generalization capability: if the training data is well selected, the performance can be improved without increasing the size of the training set. The results clearly show that the proposed subset selection technique outperforms the commonly used decimation method; the average improvement achieved using the proposed approach is 38.8%.

Figure 4 illustrates the "subset distance" (see Section 2.2) of datasets extracted using the two different data selection methods: the decimation technique and the proposed selection algorithm. It is easy to see that the average distance D is more or less even when the decimation method is used. With the proposed method, the average distance decreases monotonically with increasing data size. Furthermore, the difference between the two methods is large with small subset sizes, and converges to zero when the whole data set is used. Thus, these results indicate that the proposed method can extract data more efficiently, i.e. the selected data has better coverage. Naturally, this explains the better generalization capability of the trained model.

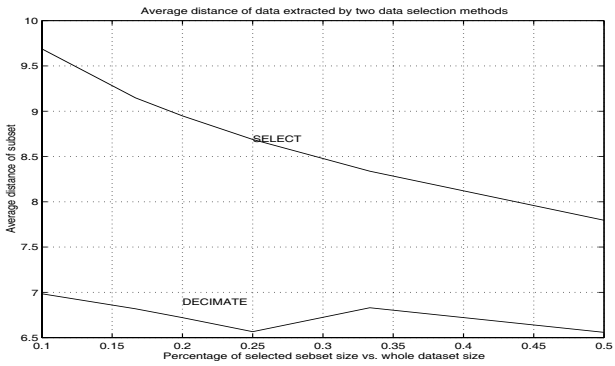

Figure 4. Average distance D inside the subsets extracted using the two different data selection methods, with respect to the percentage of the subset size vs. the whole data size.

5 CONCLUSIONS

Training data selection from a text database is a crucial, but often neglected, step in the development of ASR and TTS systems. In this paper, we define an objective function that effectively measures the quality of a selected subset. Moreover, we introduce a subset selection algorithm that optimizes the objective function. Our experimental results obtained in the syllabification task show that the proposed approach is a very promising technique that makes it possible to select subsets with good coverage in a systematic and meaningful way. The presented idea can be used in many different applications that require training with a text database.

6 REFERENCES

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

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

[3] K. Müller, "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] E. Ristad and P. Yianilos, "Learning String Edit Distance", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.20, pp.522-532, May, 1998.

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

[6] J. Suontausta, and J. Häkkinen, "Decision Tree Based Text-to-Phoneme Mapping for Speech Recognition," In *Proceedings of 6th ICSLP*, Beijing, China, 2000.

[7] J. Tian, "Data-Driven Approaches for Automatic Detection of Syllable Boundaries", in *Proceedings of 8th ICSLP*, Jeju Islands, Korea, 2004.

[8] J. Tian, J. Häkkinen, S. Riis, and K. Jensen, "On Text-Based Language Identification for Multilingual Speech Recognition Systems, In *Proceedings of 7th ICSLP*, Denver, USA, 2002.