Rapid development of a grapheme-to-phoneme system based on weighted finite state transducer (WFST) framework *

◎ Dong Yang, Paul Dixon and Sadaoki Furui (Tokyo Institute of Technology)

1 Introduction

Grapheme-to-phoneme conversion (G2P) is the task of finding the pronunciation of a word from its written format and it is a difficult task for many languages (such as English and French). G2P is an important component of speech synthesis systems, and it is also helpful in other applications, including speech recognition and language learning.

The easist way for G2P is dictionary look-up. However, this method has serious limitations: constructing a large-sized dictionary is very expensive and time-consuming, and also out-of-vocabulary words always exist. Rule-based methods have been proposed to tackle the problems of dictionary look-up. Usually a list of rules are created together with an exception list. The drawback of the rule-based method is that extracting rules is very hard and requires a large amount of linguistic knowledge.

To overcome the difficulties of the previous methods, data-driven approaches have been proposed [1, 2, 3]. [1] proposed a joint source channel method for G2P, and [2] proposed using maximum entropy model with n-gram features, while [3] used joint sequence model which uses co-sequence and grapheme-phoneme joint n-grams and it obtained state-of-the-art results. If we denote an orthographic form of a word as G, which is composed of a sequence of written units, and a pronunciation as P, composed of a sequence of phoneme symbols, the G2P problem can be written as:

$$\widetilde{P} = \arg\max_P Pr(P|G) = \arg\max_P Pr(P,G) \qquad (1)$$

This paper uses the joint source channel model [1] in a weighted finite state transducer (WFST) framework. The algorithm is not new, but our development effort shows that we can build a G2P system in a very short time by making use of publicly available toolkits. The training time can be shortened greatly without the performance decreasing, when compared to the state-of-the-art result [3].

2 Theoretic background

2.1 Joint source channel model

The second part of Equation 1 is a joint source channel model and it describes how graphemes and phonemes are generated simultaneously by the model. If the spelling of a word is $G = (g_1, g_2, ..., g_N)$, and its pronunciation is $P = (p_1, p_2, ..., p_N)$, we compute the probability according to:

$$\begin{aligned} & Pr(P,G) = Pr(p_1,p_2,...,p_N,g_1,g_2,...,g_N) \\ = \; & Pr(<g_1,p_1>,<g_2,p_2>,...,<g_N,p_N>) \\ = \; & Pr(<g,p>_1,<g,p>_2,...,<g,p>_N) \\ = \; & \prod_{k=1}^{N} Pr(<g,p>_k \mid <g,p>_1^{k-1}) \qquad (2) \end{aligned}$$

2.2 Grapheme-phoneme alignment

The alignment between graphemes and phonemes is not normally available in a dictionary. In this paper, we make use of a Viterbi version of the expectation-maximization (EM) algorithm to find the best alignment,

$$\widetilde{A} = \arg\max_A Pr(P,G,A) \qquad (3)$$

in which A is the alignment and is obtained using the following procedure:

1. Initialize a random alignment
2. Calculate n-gram probabilities from initialized data
3. E-step: apply the n-gram model to realign each entry in the dictionary
4. M-step: update n-gram probabilities from re-aligned data
5. Go to 3, until the alignment converges

3 Fast development of a G2P system

3.1 Alignment

There are several types of alignment methods: we can either group several graphemes or phonemes together to let the G and P share the same number of units; we can split graphemes or phonemes or insert null graphemes or phonemes to make G and P have the same length. In this paper, we choose to group graphemes and align the grapheme chunks to each phoneme. After achieving a converged alignment, we insert null phonemes into P to obtain

*重み付き有限状態トランスデューサを用いた書記素から音素への変換システム開発の高速化，楊 冬， ポール ディクソン， 古井 貞熙（東工大）

"one grapheme to one phoneme" correspondence. eg. (bike, /bIk/), aligned as (b i ke, /b I k/), becomes (b i k e, /b I k -/) after inserting null phoneme /-/.

3.2 WFST decoder for G2P

Our decoding approach makes use of WFSTs to represent the models and simplify the development by utilizing standard operations such as composition and shortest path algorithms.

After the alignments are generated, the first step is to build a *corpus* to train the G2P WFST. Each aligned word is converted to a sequence of grapheme-phoneme pairs $\langle g,p\rangle_1, \langle g,p\rangle_2, ... \langle g,p\rangle_k$, where each g is a character and each p is a phoneme. Each of the pairs is treated as a word and the entire set of alignments is used to train an n-gram language model. In these evaluations we used the MITLM toolkit [4] to build an n-gram model with modified Kneser-Ney smoothing.

We then use the procedure described in [5] and convert the n-gram to a weighted acceptor representation where each input label belongs to the set of G2P alignment pairs. Next the pairs labels are broken down into the input and output parts and the acceptor is converted to a transducer M.

To perform the actual G2P, the input word is converted to an acceptor I which has one arc for each of the characters in the word. I is then composed with M according to $O = I \circ M$ where $\circ$ denotes the composition operator. The n-best paths are extracted from O by projecting the output, removing the epsilon labels and applying the n-shortest paths algorithm with determinization from the OpenFst Toolkit [6].

4 Experiments

The NetTalk dictionary is publicly available and has been widely used in previous research. We choose to use this dictionary for our experiment and we try to replicate the conditions reported in previous studies[1], so that the performance figures can be compared directly. Furthermore, the NetTalk data has alignments that serve to validate the correctness of our alignment implementation. We can see in Table 1 that our system performs as well as the state-of-the-art results reported in [3]. Regarding the training time, although we cannot find enough information to exactly repeat the same experiment in [3], a simple timing experiment shows that the training takes hours to finish using the tool from [3], compared to less than one minute of our system.

[1]We would like to express our thanks to Stanley F. Chen for his kind sharing of the data partition.

Data Set	System	WER	training time
NetTalk	[3]	31.00	hours
	this work	31.04	29 seconds

Table 1 Comparing our system with the state-of-the-art system (WER:word error rate)

5 Conclusions and future work

In this paper, we presented an approach for rapidly developing a G2P tool using publicly available toolkits. Our experiment showed that our system performed as well as another state-of-the-art system. Areas for future work include improving our alignment method and performing experiments on languages other than English.

References

[1] Lucian Galescu and James F. Allen, *Bi-directional Conversion Between Graphemes and Phonemes Using a Joint N-gram Model*, Proceedings of the 4th ISCA Tutorial and Research Workshop on Speech Synthesis 2001

[2] Stanley F. Chen, *Conditional and Joint Models for Grapheme-to-phoneme Conversion*, Proceedings of European Conf. on Speech Communication and Technology 2003, pages 359-394

[3] Maximilian Bisani and Hermann Ney, *Joint-sequence models for grapheme-to-phoneme conversion*, Speech Communication 50 2008, pages 434-451

[4] Bo-June Hsu and James Glass, *Iterative Language Model Estimation: Efficient Data Structure & Algorithms*, Proceedings Interspeech 2008, pages 841-844.

[5] Diamantino Caseiro, Isabel Trancosoo, Luis Oliveira and Ceu Viana, *Grapheme-to-phone using finite state transducers*, Proceedings 2002 IEEE Workshop on Speech Synthesis.

[6] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut and Mehryar Mohri, *OpenFst: A General and Efficient Weighted Finite-State Transducer Library*, Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007), pages 11-23.