



Modeling Pronunciation of OOV Words for Speech Recognition

Zeeshan Ahmed, Julie Carson-Berndsen

CNGL, School of Computer Science and Informatics, University College Dublin, Ireland

zeeshan.ahmed@ucdconnect.ie, julie.berndsen@ucd.ie

Abstract

This paper presents a technique for modeling pronunciation in automatic speech recognition using an approach based on statistical machine translation. The task of a pronunciation model in speech recognition is to convert a sequence of phonemes into proper words of the language. This task can be realized as a machine translation approach, whereby the source language is a sequence of phonemes and the target language is a sequence of letters forming words. The model presented in this paper specially targets out-of-vocabulary words or words with several different pronunciations. A dynamic string alignment algorithm is applied to learn the phoneme-to-letter alignment. Pronunciation segments are then extracted from these alignments and are used by the decoder to recognize words from an unknown sequence of phonemes. In contrast to usual statistical machine translation decoding, the decoder presented here uses a probabilistic finite state model rather than normal n-gram language models. A number of experiments were performed using the CMU pronunciation dictionary and the results obtained are quite promising, even where only small amounts of training data are used.

Index Terms: speech recognition, pronunciation modeling

1. Introduction

The role of a pronunciation model in automatic speech recognition (ASR) is to recognize correct words in the language given the phonemes. In previous approaches, dictionary based models have been widely used. These models are tightly coupled with underlying acoustic models to maximize speech recognition accuracy. However, they cannot handle out-of-vocabulary (OOV) words which limits the ASR system.

In this paper, an approach is presented to deal with OOV words or words which have a number of different pronunciations. The approach is inspired by statistical machine translation (SMT). Here, the source language is considered to be a phoneme sequence and the target language is considered to be a sequence of letters forming valid words of a language. In this scenario, off-the-shelf SMT tools and techniques are not applied directly, but some background knowledge is used to customize the SMT principals to suit the problem needs.

Linguistic knowledge is used during the pronunciation learning phase as well as during the decoding phase. During the learning phase, consonant and vowel information is used for the purposes of alignment. During the decoding phase, probabilistic finite state model (FSM) is used instead of statistical n-gram based language model to model spellings of the words. The

⁰This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Center for Next Generation Localization (www.cngl.ie) at University College Dublin. The opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Science Foundation Ireland.

motivation underlying the use of an FSM is that it is less prone to error because it keeps complete history in context as against $n - 1$ history in n-gram language model.

The pronunciations are learned using a Dynamic String Alignment (DSA) algorithm. With the DSA algorithm, the strings are optimally aligned by calculating scores for each alignment possibility. The alignments found in this manner can be ambiguous because of the various possibilities of alignment. These ambiguities are minimized by obtaining alignment from left-to-right & right-to-left and by using the contextual information surrounding the alignment.

The paper is structured as follows. The next section outlines approaches to letter-to-phoneme alignment and issues pertaining to the problem being addressed in this paper, namely phoneme-to-letter alignment. Section 3 describes the approach for phoneme-to-letter alignment using DSA. Section 4 explains how pronunciations are extracted from alignments and section 5 illustrates the recognition/decoding process. Section 6 presents the results and conclusion is presented in section 7.

2. Letter-to-Phoneme Alignment

Automatic letter-to-phoneme alignment has been targeted from the perspective of speech synthesis for a long time. The techniques developed previously include pronunciation by analogy [1], constraint satisfaction [2], Hidden Markov Model [3], decision trees [4], and neural networks [5]. The detailed discussion on letter-to-phoneme alignment can be found in [6].

The closest approaches to the work presented in this paper were developed by [6], [7] and [8]. [6] uses DSA algorithm to align letters to phonemes by firstly using the Expectation Maximization (EM) algorithm to align letters and phonemes on a one-to-one basis. The alignment score obtained from the EM algorithm reflect the degree of association between the letter and phoneme inventories. This score is then used by the DSA to align letters and phonemes in each word.

The approach developed by [7] also uses the EM algorithm for alignment. The algorithm allows many-to-many alignment between letters and phonemes. Chunking is then performed on the word. Once the chunks are identified, a local phoneme predictor module is used to predict the phoneme for every letter in the word using viterbi search algorithm.

[8] applies a pure SMT approach to deal with letter-to-phoneme mapping for speech synthesis. This approach uses GIZA++ and A* beam search decoder to learn letter-to-phoneme alignment and to decode respectively using the MOSES toolkit [9] which implements all of these tools.

All of these alignment techniques were developed from the perspective of speech synthesis. In this paper, the alignment problem is similar to letter-to-phoneme alignment but is concerned with the other direction, namely phoneme-to-letter. The phoneme-to-letter conversion has been successfully applied by

(a) Left to Right Alignment (b) Right to Left Alignment

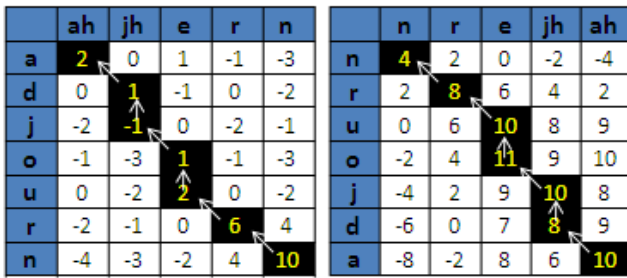


Figure 1: (a) Alignment Matrix for Left-to-Right alignment. (b) Alignment Matrix for Right-to-Left alignment.

Decadt [10] for dealing with OOV words in speech recognition. He used Memory Based Learning technique for phoneme-to-letter conversion. His work can not be directly compared with this work due to the fact that a different dataset was used.

3. Dynamic String Alignment (DSA)

To align phonemes and letters, the DSA algorithm is employed. The DSA algorithm uses the Dynamic Programming (DP) [11] technique to optimally align two string sequences. The technique for DSA in this paper is similar to [6]. However, instead of using EM algorithm to drive letter-phoneme association matrix A , basic linguistic knowledge about the pronunciation of words is used. The scoring table is created manually based on the classification with respect to vowels and consonants for a language. For the case of English, the scoring table is shown in Table 1. The Table 1 replaces the association matrix A used by the Damper DSA algorithm.

Exact Match	4
Vowel to Vowel	2
Consonants to Consonants (ambiguous mapping like c and k etc.)	2
No match	-1

Table 1: Alignment Scoring

In this table, *Exact Match* means the case when a phone like /p/ matches "p", /b/ matches "b", /k/ matches "k" etc. The score is 2 when ambiguous consonants like /k/ matches "c", /s/ matches "z", /l/ matches "ph" etc. For the vowel to vowel matching, the score is also 2. If there is no match, the score is -1. All of these scores have been obtained based on various experiments with different settings. These scores produced the best results for the experiments.

Two alignments are generated between the letter and phoneme sequence of each word; one from left-to-right and other from right-to-left as shown in Figure 1. This is because sometimes left-to-right and right-to-left alignment generate slightly different alignments which are then merged to get a final alignment. The alignments are merged using a merge alignment operation. If the alignments in left-to-right and right-to-left are the same, no merge operation is required. For instance, in Figure 2, /ah/ → a, r → r and n → n are the same in both alignments therefore no merge operation is needed. In contrast, the alignment /jh/ → j and /null/ → d in left-to-

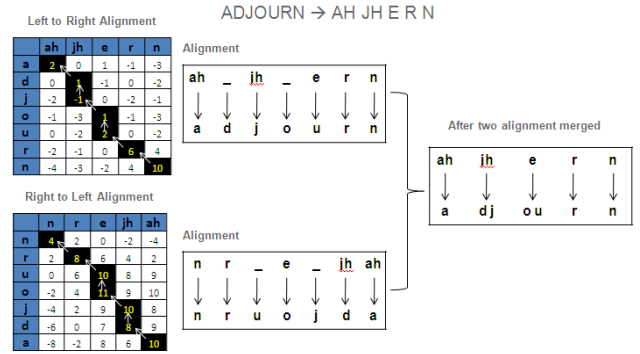


Figure 2: Dynamic String Alignment. Left-to-right, right-to-left and final alignment after merged operation

right and /jh/ → d and /null/ → j in right-to-left are different. In the merge operation, these four alignments are merged in single alignment as /jh/ → dj, similarly with /e/ → ou. Figure 2 depicts the overview of the merge operation.

4. Pronunciation Extraction

The pronunciation extraction process extracts the segments of the alignments that can be re-used. Note that segment here refers to groups of phonemes or letters. These extracted segments of word pronunciation are then used during the decoding process to recognize words. This step is necessary because *null* alignments are one of the biggest hurdles in using alignment directly for recognition. For instance, consider the following alignment for English word *backed*.

b	ae	-	k	-	t
b	a	c	k	e	d

Here, the letter "e" surrounded by letters "k" and "d" is silent, and the letter "c" surrounded by "a" and "k" is also silent. It is very difficult during decoding phase to translate *null* phoneme alignments. The objective of this step is thus to analyze the alignment generated from the previous step and automatically produce the pronunciation segments that can be re-used.

The pronunciation extraction step analyzes every ambiguous alignment and takes the left and right alignment (context) to disambiguate the alignment. An ambiguous alignment is a non-exact, *null*, 1-to-many, many-to-1, many-to-many and vowel-to-vowel alignment. The pronunciation segments generated as a result of this step include the alignment as well as the context of the alignment i.e. left or right alignment or both. If the left and right context are still ambiguous, the segment is further extended to left and right until the ambiguity is resolved. The right context may be ignored, or not considered the part of the segment, if the immediate alignment to its right is also ambiguous. Finally, all the pronunciation segments are saved in a pronunciation table. The pronunciation table contains the segment as well as translation probability of each segment. The probability is calculated using maximum likelihood estimation (MLE) by counting the occurrence of the segment in the whole dictionary.

Figure 3 depicts two examples of how pronunciations are extracted. Apart from the extracted segments, all the exact alignment as well as vowel-to-vowel alignments are also in-

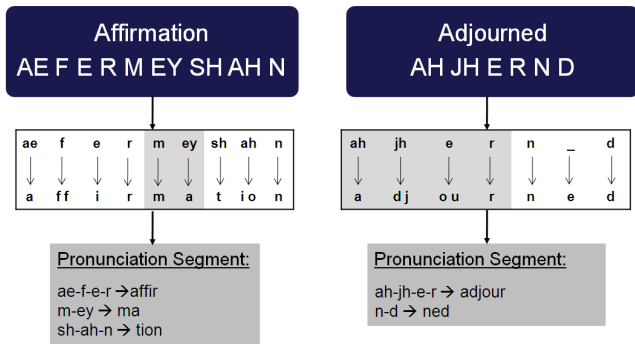


Figure 3: Pronunciation Extraction

cluded in pronunciation table. This is done to deal with OOV words during the recognition phase.

5. Word Recognition

The following is the maximization equation for the decoding process:

$$w' = \arg \max_w \{Pr(w) * Pr(p|w)\}$$

where, $Pr(p|w)$ is the probability of translating the phoneme sequence p into the word w in pronunciation model and $Pr(w)$ is probability of the word w in the word model. The decoding approach applied is similar to stack-based decoding as described in [12].

The decoder uses a word model in the form of probabilistic finite state machine. This model is like the prefix tree representation of the word spelling as shown for the three-word English dictionary in Figure 4 (b). Here, the letters of the word form states in the FSM and the transitions between these letters represent next possible letters in the word. All the words which have the same prefix have the same previous states as shown in Figure 4 (b). There is a default start state \$ for each word. The accepting states are highlighted in dark. The reason for using an FSM is that it is less prone to error as compared to an n-gram model. In an n-gram model, it is difficult to capture the integrity or linguistic well-formedness of the word. The FSM model, on the other hand, can easily model word spellings as well as protect the integrity of the linguistic structure. Furthermore, a word model is not tightly integrated with pronunciations of words which facilitates the the freedom of adding new words (without pronunciation) to the vocabulary on the fly.

Figure 4 and 5 is an illustration of a working example of the complete phoneme-to-letter translation for the English word *LIGHTER* using the approach described above. This example is simple, and for illustrative purposes only, but it highlights some of the main concepts underlying the approach presented in this paper. Figure 4 (a) and (b) show the pronunciation model for a two-word only pronunciation dictionary and a word model for a three-word dictionary respectively. Figure 5 depicts the decoding/recognition process.

6. Evaluation

The pronunciation modeling approach has been applied on the English CMU Pronunciation Dictionary CMUDict¹. Table 2

¹<https://cmusphinx.svn.sourceforge.net/svnroot/cmusphinx/trunk/cmudict/>

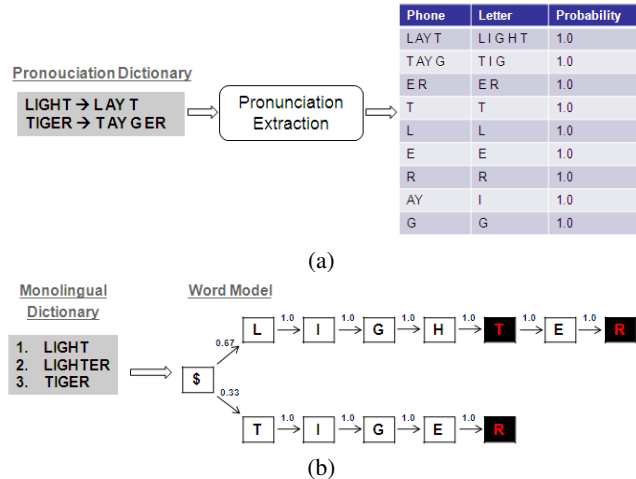


Figure 4: (a) Pronunciation model extraction example for two-word pronunciation dictionary. (b) Word model example for three-word dictionary.

Decoding Process

LAYER = ???

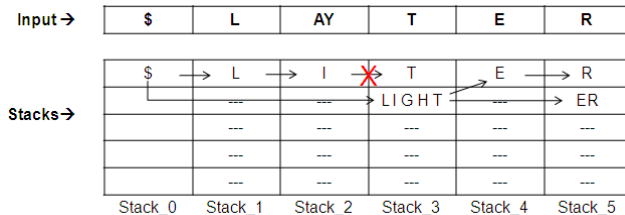


Figure 5: Word recognition process using stack decoding algorithm. The algorithm uses the pronunciation and word model from Figure 4. The cross sign indicates that transition from "I" to "T" is not possible in the word model.

shows the overall statistics of the CMUDict data.

In CMUDict, abbreviations are also present. The abbreviations are eliminated from dictionary during evaluation; this is done in order to remove erroneous pronunciation segments and thus improve the word recognition accuracy. In order to eliminate abbreviations, a rule is used such that, if the number of phoneme in the phoneme sequence of a word is greater than double the size of a word in terms of letters, the word is ignored during learning phase. Moreover, stress markers are also removed during learning phase. In the experiment, the phoneme /ER/ was found to be very confusing during pronunciation learning phase due to the fact that /ER/ is a combination of two phonemes; the vowel /E/ and the consonant /R/. Since vowel/consonant distinctions are used to align phonemes and letters, the /ER/ phoneme does not fit into either category. The /ER/ phoneme is thus split into two separate phonemes /E/ and /R/ for better accuracy.

The evaluation strategy is *Word Correctness* i.e. the percentage(%) of words recognized correctly w.r.t to a reference word. The experiment for an N-best list is also performed. In this case, the N-best recognized words are compared with ref-

Words	133358
Phoneme	84
Letter	26
No. of pronunciation of single word	max. 6
No. of words with same pronunciation	max. 13
Words pronounced at least in 2 different ways	10696

Table 2: CMUDict Data Statistics

S.No	Train - Test	Training Words	Testing Words
1	90%-10%	120022	13335
2	50%-50%	66680	66679
3	10%-90%	13335	120022

Table 3: Experiment Dataset

erence word. If any of the words in the N-best list matches the reference word then it is marked as correct. The purpose of producing an N-best list is for further use in a sophisticated language model for continuous speech recognition.

Table 4 presents the results of the experiment. For each experiment, 5-fold cross validation scheme has been performed and average of these five results are represented in this table. The % of correct word recognitions obtained for 1, 5, 10 and 50 best words are presented. It is important to note that results of the three datasets do not differ according to the size of the data sets. This demonstrates that this approach is applicable given even a small amount of training data.

In CMUDict, the maximum number of words with the same pronunciation is 13 and number of words that can be pronounced at least in two different ways are 10696. This is the natural ambiguity present in pronunciation of a language and therefore, a single best recognition result is not sufficient. For this reason, the N-best list results in Table 4 are better than the single best result. Another reason underlying the significant difference between single best and N-best results is that the model deals with noisy inputs which sometimes results in incorrect words being more highly ranked than correct words. The FSM word model plays an important role in dealing with noisy inputs and effectively eliminates incorrect words during recognition.

7. Conclusions

This paper has presented a novel pronunciation modeling approach for phoneme-to-letter conversion in ASR to deal with OOV words. The approach employs principles of statistical machine translation. Pronunciation alignments are learned using DSA algorithm and are further refined using a pronunciation extraction process. The decoding algorithm has been revised to use the pronunciation model and word model for phoneme to letter conversion. A word model has been developed in the form of prefix tree spelling model using FSM in contrast to usual n-gram language model in SMT. The alignment learning approach presented is novel in the sense that phoneme-to-letter alignment using DSA uses very basic linguistic knowledge which makes this approach usable even for small amounts of training data as reflected in experimental results.

As future work, we plan to use presented technique in more sophisticated language model to develop a language modeling toolkit for continuous speech recognition.

S.No	Train - Test	1-Best	5-Best	10-Best	50-Best
1	90%-10%	65.74	90.63	95.05	98.02
2	50%-50%	66.19	90.33	94.45	97.11
3	10%-90%	65.54	86.52	90.06	92.09

Table 4: Results of Pronunciation Modeling approach when allied to CMUDict.

8. References

- [1] Y. Marchand and R. I. Dampier, "A multistrategy approach to improving pronunciation by analogy," *Computational Linguistics*, vol. 26, no. 2, pp. 195–219, 2000.
- [2] A. van den Bosch and S. Canisius, "Improved morpho-phonological sequence processing with constraint satisfaction inference," in *SIGPHON 2006: Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology and Morphology*. Morristown, NJ, USA: Association for Computational Linguistics, 2006, pp. 41–49.
- [3] P. Taylor, "Hidden markov models for grapheme to phoneme conversion," in *Proceedings of the 9th European Conference on Speech Communication and Technology*, 2005.
- [4] A. W. Black, K. Lenzo, and V. Pagel, "Issues in building general letter to sound rules," in *The Third ESCA Workshop in Speech Synthesis*, 1998, pp. 77–80.
- [5] T. J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce english text," in *Complex System*, 1987, pp. 145–168.
- [6] R. I. Dampier, Y. Marchand, J.-D. Marsters, and A. Bazin, "Aligning letters and phonemes for speech synthesis," *5th ISCA Speech Synthesis Workshop*, 2004.
- [7] S. Jiampojarn, G. Kondrak, and T. Sherif, "Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion," in *HLT 2007: The Conference of the NAACL; Proceedings of the Main Conference*. Rochester, New York: ACL, April 2007, pp. 372–379.
- [8] T. Rama, A. K. Singh, and S. Kolachina, "Modeling letter-to-phoneme conversion as a phrase based statistical machine translation problem with minimum error rate training," in *Proceedings of the NAACL HLT Student Research Workshop and Doctoral Consortium*. Boulder, Colorado: ACL, June 2009, pp. 90–95.
- [9] P. Koehn, A. B. Hieu Hoang, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *Annual Meeting of the Association for Computational Linguistics (ACL), demonstration session*, Prague, Czech Republic, June 2007.
- [10] B. Decadt, J. Duchateau, W. Daelemans, and P. Wambacq, "Phoneme-to-grapheme conversion for out-of-vocabulary words in speech recognition," 2001.
- [11] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University, 1957.
- [12] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM Journal of Research and Development*, vol. 13, pp. 675–685, 1969.