HMM-based Pronunciation Dictionary Generation

Arthur Kantor[1], *Mark Hasegawa-Johnson*[2]

[1]Department of Electrical & Computer Engineering, University of Illinois at Urbana-Champaign, USA
[2]Department of Computer Science, University of Illinois at Urbana-Champaign, USA
{akantor, jhasegaw}@illinois.edu

Abstract

In this paper, we discuss automatically generating a phonetic pronunciation from an orthographic spelling of words. The letter-sequence to phoneme-sequence mapping is useful in a variety of contexts, including text-to-speech applications, automatic spelling correction, and generating a pronunciation lexicon for a new training dataset which contains out-of-vocabulary words. A system based on hidden Markov models is described, and is then used to generate pronunciations for out-of-vocabulary the words and word fragments in the Fisher conversational telephone speech corpus. The Fisher phonetic pronunciations are analyzed to show that for conversational speech and a typical phonetic dictionary, a large amount of lexical ambiguity remains even when the word boundaries and phonetic transcriptions are known.

Index Terms: automatic pronunciation dictionary generation, letter-to-phone string mapping

1. Introduction

This paper concerns itself with a *pronunciation function* which maps an arbitrary string of letters to a pronunciation represented by a string of phonemes. The string of letters is usually a word or a word fragment, and the phoneme string consists of phonemes from some externally defined phoneme set. For some languages such as Russian or Czech the function can be simple mapping each letter to a phoneme string in a (mostly) context independent way. For other languages such as English or French, the pronunciation of a particular letter greatly depends on the neighbors of that letter and the pronunciation function needs to be more complicated. Such a pronunciation function is useful in a variety of tasks, including text-to-speech applications, automatic spelling correction, and generating a pronunciation lexicon for a new training dataset which contains out-of-vocabulary (OOV) words.

A variety of methods have been used to design a pronunciation function. In [1], Fisher describes a system which learns a set of context-dependent letter-to-phoneme rules. Contexts of up eight letters are used and a single rule can output multiple phonemes. The system is used to generate OOV words for a new lexicon. The paper also gives a more extensive bibliography of related work.

A similar but improved pronunciation model is used in [2] to correct spelling mistakes in the cases where the typist makes no typos but does not know the proper spelling of a word. In this case, the mistyped word will be pronounced similarly to the desired word, and looking for correctly-spelled alternatives in the pronunciation space is beneficial.

The above pronunciation functions (and the new one presented in this paper) can be made language independent, requiring only an existing dictionary for training. They can be used for completely new word generation, as well as for determining a pronunciation for fragments of existing words, which are common in conversational speech transcriptions.

In Section 2, we present a new pronunciation function which is based on hidden Markov models (HMMs) and we evaluate its accuracy in Section 3. In Section 4, we measure the coverage of OOV words in conversational speech and use our pronunciation function to generate the pronunciations for OOV words and word fragments in the Fisher corpus. The lexical ambiguity which remains even after the phonetic transcriptions are known is discussed in Section 5, and conclusions and future work directions are presented in Section 6 .

2. HMM-based letter-to-phone mapping

In this section we propose a new pronunciation function based on HMMs. We model each context-dependent phoneme (triphone) as an HMM and treat the letters as observations generated by the HMM. The HMM structure representing each triphone is shown in Figure 1. The HMM is greatly constrained: there are no self loops, and it can emit exactly one, two or three letters.

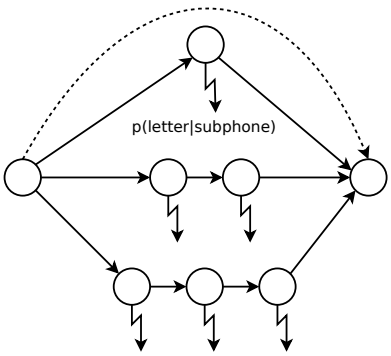

Figure 1: The finite state automaton view of an HMM representing a single triphone. Only the sub-triphone states with lightning arrows emit letters. They emit exactly one letter since there are no self loops. The dashed transition is only possible in a tee-model, and if that transition is taken, no letters will be emitted by the phoneme.

The model is iteratively trained with the expectation-maximization (EM) algorithm. The HMMs are flat-start initialized and EM is used train monophones. The triphone HMMs are created by cloning the appropriate monophone HMM and are again iteratively re-estimated with the EM algorithm. There are no shared distributions among any of the sub-triphone states (i.e. no parameter tying is performed).

No triphone clustering is performed, because we believe that our dictionary contains sufficient training data to give reasonable performance. 90% of the words in the dictionary contained 98.1% of the triphones observed within the entire dictionary, so a new word is likely to be made up of triphones already seen in the training data.

Decoding is performed with the viterbi algorithm, which uses a smoothed bigram model to specify the distribution over phoneme sequences. Viterbi search takes place over a small state space, and so search beam pruning is not necessary.

There are two advantages of the HMM based system over the rule-based system presented in [1]. One is that no initialization mapping from letters to phonemes is necessary in our system, and so no linguistic knowledge is required. The other advantage is that in the HMM system it is possible to place a prior distribution over the phoneme strings, while the rule-based system has no such prior. (However, in [2], A trigram model was used to rescore an n-best list generated by the rule-based system.)

2.1. Tee Models

In some languages (such as English) a single letter (such as `U` or `X`) may occasionally be pronounced as two phonemes, which is disallowed by our HMM structure. To solve this, an HMM can be modified into a *tee-model*, where the entire HMM can be optionally traversed without emitting even a single letter. We have implemented the entire system in the HTK toolkit [3], which allows tee-models, but does not allow two consecutive tee-models either during training or decoding.

We can consider the triphones as weighted nodes in a graph, where the nodes are adjacent if the triphones are neighbors, and node weights are the number of times the node co-generates a single letter with other nodes. Then, selecting the set of non-adjacent triphones with maximum total weight is exactly the well-known *maximum weighted independent set* problem [4], which is NP-hard.

Instead of finding the optimal solution, we select a set of tee-models greedily. We consider all the words that have more phonemes than letters, which must contain letters mapping to more than one phoneme. We make a list of all the triphone models occurring in these words, and sort the list by number of occurrences. Then we simply go down the list and tee any model which does not conflict with any of the models teed so far. This teeing of the HMM models reduces the chances of introducing spurious phonemes into a transcription of a letter string, but does not completely eliminate it. It can be combined with other approaches described in Section 6.1, and if all else fails it can back off to the pronunciation function using monophone HMMs.

The HMM teeing takes place after the triphones are cloned from monophones, and before they are re-estimated with the EM algorithm.

3. Evaluation of letter-to-phone mapping

To evaluate our approach, we train our pronunciation function on a random 90% subset of the multi-pronunciation CMU dictionary [5], and test it on the remaining 10%. The alphabet is the standard 26 english letters, plus the apostrophe and the phoneme set is the 39-phoneme set used in CMU dictionary with all stress information stripped from it.

The accuracy is evaluated in terms of the Phone Error Rate (PER) by comparing the hypothesized pronunciation to the reference pronunciation taken from the CMU dictionary. Because the CMU dictionary contains multiple pronunciations, it may happen that one pronunciation of the word ends up in the training set, and the other ends up in the test set.

The results for our monophone and triphone systems are presented in Table 1.

system	**PER**
monophone	28.5%
triphone	14.2%
best result in [1]	≈6%
best result in [2]	8.5%

Table 1: The phone error rate of our pronunciation function with monophone and triphone HMM models, along with the best results in related work.

These PERs are not completely comparable because different dictionaries were used. For example, only the subset of words that was identically pronounced in 4 out of the 11 available dictionaries was used for training and evaluation in [1] to ensure a high quality training dictionary. Some ways to make our system more competitive and ways to combine our approach with those presented in [1, 2] are described in Section 6.1.

4. Lexicon generation for the Fisher transcriptions

We use the above pronunciation function to generate phonetic pronunciations for OOV words encountered in the Fisher transcriptions which are then used in the training of a conversational speech recognizer. The Fisher corpus is a ≈2000 hour corpus of conversational telephone speech automatically segmented into utterances, with each utterance transcribed by non-expert humans. The transcriptions contain many non-speech sounds and word fragments. The multi-pronunciation CMU dictionary was used as the starting dictionary and the OOV words are the ones missing from this dictionary.

A closed set of non-speech sounds (e.g. laughter, coughing, etc...) is also transcribed and is enclosed in [ ] brackets. Word fragments are transcribed, with a - marking the missing part either in the beginning or the end of the word (e.g. `IN-` might be the first part of the word `INDIVIDUAL`).

It is not obvious how a word fragment should be pronounced: `BI-` in `BIT` is different from `BI-` in `BITE`. However, word fragments are commonly involved in repetition disfluencies, and the full word is often repeated completely within a few words of the word fragment. In that case it is usually reasonable to repair the word fragment as part of the nearby full word. Letting *nearby* mean to be within 6 words seems to give reasonable results. The word fragments in the transcription of the entire corpus were repaired this way.

Coverage statistics for all of the above cases are given in Table 2. Even though word fragments and OOV words make up 49% of the words encountered in Fisher transcriptions, they cover only 1.2% of the tokens if we exclude the 11 non-speech 'words'. The missing words and fragments are not likely to be included in a recognition vocabulary. However, they can be used in training if the phonetic transcriptions for these words were generated.

The missing pronunciations were automatically generated in one of the following ways, in the order of preference:

1. If the word is in the CMU dictionary, use that pronunciation.

	words	tokens	coverage
all words in corpus (including uncertain words)	79742	21905137	100.0%
singleton words	32703	32703	0.1493%
all words not in the CMU dictionary	39311	822146	3.7532%
total non-speech markers (enclosed in [ ] e.g. [LAUGH])	11	559629	2.5548%
word fragments (starting or ending in -)	21552	153098	0.6989%
repaired word fragments		101550	0.4636%
whole words not in CMU dictionary	16654	103244	0.4713%
multi-words not in CMU dictionary (e.g. ANTI-FRENCH)	1094	6354	0.0290%

Table 2: Coverage for transcriptions of the Fisher corpus with word fragments repaired.

2. Otherwise, if the word is an acronym or contains numbers, do direct letter-to-phones replacement, so `401K` ⇒ `F AO R OW W AH N K EY` and `I_B_M` ⇒ `AY B IY EH M`.
3. Otherwise, if the repaired word fragment in the CMU dictionary, determine the fragment pronunciation by using the HMM model to force-align the letters of the repaired word against it's phonetic pronunciation.
4. Otherwise, attempt to use HMM model to decode the phonetic pronunciation.
5. If all of the above fail, manually phonetically transcribe the word.

As Table 2 shows, of the 1.2% of the tokens needing to be transcribed, we were able to force align words covering 0.46% of the transcription - a task which can be performed accurately by our model (although we could not evaluate the force-alignment accuracy for lack of reference pronunciations).

Only 134 words reached step 5 above, and a manual transcription was required, either because some illegal alphabet was used (e.g. `M&M`) or the HMM model was unable to find a suitable sequence of hidden phonemes, because there were more letters than phonemes and skips were impossible (e.g. `C_D-ROM`).

This system trained on the full CMU dictionary is available on the web [6]. From a cursory check, forced alignment is almost always correct, while decoding works mostly, but fails in some cases where a single letter is observed from multiple phones (e.g. `X` in `ASPHYXIATES` → `/AH S F AY K AY AH T S/`).

5. Lexical perplexity given a phonetic transcription

The dictionary contains different words which are pronounced as an identical sequence of phonemes. These words are homophones in the specific pronunciation representation used by the recognizer - they need not be homophones in a more nuanced pronunciation representation. The speech recognizer will have problems with homophones, relying only on the language model to distinguish between them. In this section, we measure perplexity seen by the recognizer on conversational speech due to homophones when using a typical multi-pronunciation phonetic pronunciation dictionary.

For the Fisher corpus, the perplexity of word W given phoneme sequence S is calculated from the transcriptions as follows. Let $p(W|S = s)$ be the probability of a token being the word W, given that the token is pronounced as s. This can be estimated from the data as

$$p(W|S = s) \simeq \frac{C(W)}{C(s)}$$

where $C(W)$ is the count of tokens of word W, and $C(s)$ is the count of all words pronounced as s. This works if each word has a single pronunciation.

If a word W has M pronunciations, and occurs in the corpus N times, we assume a uniform distribution over pronunciations, so W is pronounced with any given pronunciation $\lfloor \frac{N}{M} \rfloor$ times in the corpus. Using a uniform distribution here is suboptimal, as the pronunciations per word tend to have something like a Zipf's law distribution [7], but certainly not a uniform distribution. However, we have no basis to make a better guess without phonetically transcribing the tokens for the multi-pronunciation words. The word perplexity given a pronunciation calculated with this assumption will over-estimate the true word perplexity. 29.85% of the tokens have pronunciations that could have been generated by a multi-pronunciation word. These are the tokens affected by the 'uniform distribution over multiple pronunciations' assumption.

Let $H(W|S = s)$ be the entropy (in bits) of the distribution $p(W|S = s)$. Then

$$H(W|S) = E_{p(S)}[H(W|S = s)]$$

where

$$p(S) = \frac{C(S)}{\sum_S C(S)}$$

$H(W|S)$ was estimated using the multi-pronunciation dictionary and the Fisher corpus with word fragments repaired as described in the previous section. The *conditional lexical perplexity* of the word given the phonetically transcribed token is computed to be

$$2^{H(W|S)} = 1.17$$

words per phonetically pronounced token in the Fisher corpus.

A conditional lexical perplexity of 1.17 is equivalent to 17% of the tokens from typical conversational speech having two pronunciations. The semantics of the conversation are important for identifying those ambiguous 17% of tokens, but there are also some indications that uncaptured acoustic distinctions do exist. For example, words having the same phonetic transcriptions (such as `KNOW` and `NO`) differ in average duration depending on the identity of the word[8]. The above observation combined with the high conditional lexical perplexity is a motivation to explore longer-duration, more specialized pronunciation units than the typical phones and triphones. It is also a motivation to consider acoustic features with durations longer than the 25ms PLPs and MFCCs commonly used in recognition. These issues are explored in more detail in [9].

6. Conclusions

We presented a pronunciation function mapping letter strings to phonetic pronunciations of those strings and applied it to gener-

ate the missing pronunciations for words in the Fisher conversational telephone speech corpus. This tool is language independent and may be useful for others working with new speech datasets.

We've also explored the lexical perplexity of the word given the phonetically transcribed token, and shown that the conditional lexical perplexity is quite high for conversational speech. This implies that a phonetic pronunciation dictionary using a typical phoneme set does not capture all of the information to differentiate between distinct words.

Although no direct comparison was made, our proposed pronunciation function probably underperforms the approaches suggested previously. However, there are some easy ways to improve the quality of our pronunciation function, and more interestingly it is possible to combine our pronunciation function with the successful previous approaches.

6.1. Future work

One easy way to improve the system is to use n-grams with $n > 2$ in the "language model" defining distribution of phoneme strings. Another improvement is to use quinphone HMMs instead of triphone HMMs and to cluster the quinphone HMMs when their conditional distributions are sufficiently similar. Using quinphones instead of triphones would not only sharpen the observation distributions of the HMMs, but will also reduce the fraction of adjacent teed models, further improving model quality.

In English, only some letters (e.g. `U` and `X`) are often pronounced with more than one phoneme, so an easy way further reduce adjacent teed models is by splitting those problematic letters into two parts (e.g. $\texttt{U}_1$ and $\texttt{U}_2$) and ensure that they follow each other. This approach is language-specific, however.

As third way to reduce the effect of adjacent teed models, we can use one of the many better optimization algorithms for picking a high-weight weighted independent set [4].

Another interesting improvement would be to combine the approaches in [1, 2] with the HMM-based pronunciation function described here in a way similar to a Hybrid HMM. A distribution for

$$p(phonemeSequence|letter_{i-n}, ..., letter_i, ..., letter_{i+n}) \quad (1)$$

can be estimated as in [1, 2]. Most of the time, $phonemeSequence$ consists of only a single phoneme and in this case it could be provided as virtual evidence to the triphone (or quinphone) system described here. In the rare cases where $phonemeSequence$ with multiple phonemes has significant probability, an alternate pronunciation with more phonemes could be hypothesized, weighed by the probability given by Equation 1 and fed into our pronunciation function. This kind of top-down and bottom-up hybrid approach handles interdependencies between neighboring phonemes and between neighboring letters, and could potentially give even better accuracy than either approach alone.

Another direction for future work is to gather statistics on which kinds of letter sequences decode to wrong phonetic sequences. Answering questions like "Are letter sequences from foreign words mispronounced?", or "Is multiple phones per letter the main problem?" may lead to further improvement of the pronunciation function.

Finally, there is a question about the minimum achievable phone error rate (PER) of the pronunciation function. Should the word `EXCITE` be pronounced as `/IH K S AY T/` which is suggested by the CMU dictionary or `/EH K S AY T/` as suggested by our HMM pronunciation model? It would be nice to establish some sort of lower bound on the PER of any pronunciation function from the pronunciations in the training dictionary.

7. References

[1] W. M. Fisher, "A statistical text-to-phone function using ngrams and rules," in *in ICASSP*, 1999, pp. 649–652.

[2] K. Toutanova and R. C. Moore, "Pronunciation modeling for improved spelling correction," in *Proceedings of the Association for Computational Linguistics*, 2002, pp. 144–151.

[3] J. Odell, D. Ollason, P. Woodland, S. Young, and J. Jansen, *The HTK Book for HTK V2.0*. Cambridge University Press, Cambridge, UK, 1995.

[4] P. M. Pardalos and J. Xue, "The maximum clique problem," *Journal of Global Optimization*, vol. 4, pp. 301–328, 1994.

[5] "Carnegie-Mellon Pronouncing Dictionary," http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

[6] A. Kantor, "Letter-to-Phone String Mapping Tool," http://mickey.ifp.uiuc.edu/speech/webpronounce/webpronounce.cgi.

[7] S. Greenberg, "Speaking in shorthand – a syllable-centric perspective for understanding pronunciation variation," in *ESCA Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition*, 1998.

[8] A. Ganapathiraju, J. Hamaker, J. Picone, M. Ordowski, and G. R. Doddington, "Syllable-based large vocabulary continuous speech recognition," *IEEE Transactions On Speech and Audio Processing*, 2001.

[9] A. Kantor, "Pronunciation Modeling For Large Vocabulary Speech Recognition," Ph.D. dissertation, University of Illinois, Urbana-Champaign, 2010.