An MDL-based approach to extracting subword units for grapheme-to-phoneme conversion

Sravana Reddy
Department of Computer Science
The University of Chicago
Chicago, IL 60637
sravana@cs.uchicago.edu

John Goldsmith
Departments of Linguistics
and Computer Science
The University of Chicago
Chicago, IL 60637
goldsmith@uchicago.edu

Abstract

We address a key problem in grapheme-to-phoneme conversion: the ambiguity in mapping grapheme units to phonemes. Rather than using *single* letters and phonemes as units, we propose learning chunks, or *subwords*, to reduce ambiguity. This can be interpreted as learning a lexicon of subwords that has minimum description length. We implement an algorithm to build such a lexicon, as well as a simple decoder that uses these subwords.

1 Introduction

A system for converting written words to their pronunciations is an important component of speech-related applications, especially in large vocabulary tasks. This problem, commonly termed "grapheme-to-phoneme conversion", or g2p, is non-trivial for several written languages, including English, since a given letter (grapheme) may represent one of several possible phonemes, depending on the context. Because the length of the context varies throughout the dictionary, fixed-length contexts may overfit some words, or inaccurately model others.

We approach this problem by treating g2p as a function from *contiguous sequences* of graphemes, which we call 'grapheme subwords', to sequences of phonemes ('phoneme subwords'), so that there is *minimal ambiguity* in finding the phoneme subword that corresponds to a given grapheme subword. That is, we seek to minimize both these quantities:

1. The conditional entropy of the phoneme subwords given a grapheme subword. This directly tackles the problem of ambiguity – a perfectly unambiguous phoneme subword conditional distribution would have entropy = 0.
2. The entropy of the grapheme subwords. This prevents the model from getting arbitrarily complex.

As a toy example, consider the following word-pronunciation[1] pairs:

time	T AY M
sting	S T IH NG
negation	N AH G EY SH AH N

There are at least 5 graphemes whose corresponding phoneme distribution is ambiguous ('i', 'e', 't', 'n', 'g'). In the segmentation below, every grapheme subword corresponds to only one phoneme subword:

t + ime	T + AY M
s + t + ing	S + T + IH NG
neg + a + tion	N AH G + EY + SH AH N

2 Related Work

Many grapheme-to-phoneme algorithms rely on something resembling subwords; these are mainly used to account for sequences of letters representing a single phoneme ('ph' for F), or vice versa ('x' for K S). Some of the early works that create one-to-one alignments between a word and its pronunciation address these cases by allowing a letter to map to one phoneme, a null phoneme, or 2-3 phonemes.

Jiampojamarn and Kondrak (2009) use expectation-maximization (EM) to learn many-to-many alignments between words and pronunciations, effectively obtaining subwords.

[1] All phonemes are denoted by their Arpabet representations.

Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, pages 713–716,
Los Angeles, California, June 2010.

Joint-sequence models divide a word-pronunciation pair into a sequence of disjoint *graphones* or *graphonemes* – tuples containing grapheme and phoneme subwords. Such segmentations may include only trivial graphones containing subwords of length at most 1 (Chen, 2003). Other such models use EM to learn the maximum likelihood segmentation into graphones (Deligne and Bimbot, 1995; Bisani and Ney, 2008; Vozilla et al., 2003).

Subwords – or phrases – are used widely in machine translation. There is a large body of work on phrase extraction starting from word alignments; see Koehn et al. (2003) for a review. Marcu and Wong (2002) learn phrases directly from sentence pairs using a joint probability model.

3 Subword Extraction

3.1 Motivation for using MDL

Consider a lexicon of grapheme subwords $\mathcal{G}$ and phoneme subwords $\mathcal{P}$ that is extracted from a dictionary of word-pronunciation pairs, along with a joint probability distribution over $\mathcal{G}$ and $\mathcal{P}$. As stated earlier, our objective is to minimize the entropy of phoneme subwords conditioned on a given grapheme subword, as well as the entropy of the grapheme subwords. That is, we would like to minimize $H(\mathcal{P}|\mathcal{G}) + H(\mathcal{G})$, which is

$$H(\mathcal{G}, \mathcal{P}) = -\sum_{g \in \mathcal{G}} \sum_{p \in \mathcal{P}} pr(g, p) \log pr(g, p) \quad (1)$$

This objective can be restated as minimizing the *expected description length* of the lexicon, which is given by its entropy. This is reflected in the MDL principle (Rissanen, 1978), which seeks to find a lexicon such that the description length of the lexicon (and the compression of the data under the lexicon) is minimized.

3.2 Lexicon Induction

We begin with an initial alignment between a word's graphemes and the phonemes in its pronunciation for all word-pronunciation pairs in the training dictionary. These alignments are derived using the standard string edit distance dynamic programming algorithm (Wagner and Fischer, 1974), giving a list of tuples $t = [(w_1, r_1), (w_2, r_2), \ldots]$ for each word-pronunciation pair.[2] The set of all tuple lists t composes the training dictionary $\mathcal{T}$.

The initial lexicon is composed of all singleton graphemes and phonemes (including null). The probability $pr(g, p)$ is taken to be the number of times the tuple (g, p) occurs in $\mathcal{T}$ divided by the total number of tuples over all alignments in $\mathcal{T}$.

Following a procedure similar the word-discovery algorithm of de Marcken (1996), the lexicon is iteratively updated as sketched in Table 1. At no point do we delete singleton graphemes or phonemes.

The subwords in the final updated lexicon are then used to decode the pronunciations of unseen words.

4 G2P Decoding

4.1 Joint segmentation and decoding

Finding the pronunciation of a word based on the induced subword lexicon involves segmenting the word into a sequence of grapheme subwords, and mapping it to a sequence of phoneme subwords.

One possibility is carry these steps out sequentially: first parse the word into grapheme subwords, and then use a sequence labeling algorithm to find the best corresponding sequence of phoneme subwords. However, it is likely that the true pronunciation of a word is *not* derived from its best parse into grapheme units. For example, the best parse of the word 'gnat' is 'g nat', which yields the pronunciation `G N AE T`, while the parse 'gn at' would give the correct pronunciation `N AE T`.

Therefore, we search for the best pronunciation over *all segmentations* of the word, adapting the monotone search algorithm proposed by Zens and Ney (2004) for phrase-based machine translation.[3]

4.2 Smoothing

A bigram model is used over both the grapheme and phoneme subwords. These bigrams need to be smoothed before the decoding step. Adding an equal probability mass to unseen bigrams would fail to reflect simple phonotactics (patterns that govern sound

[2] Phoneme insertions and deletions are represented by the null grapheme and null phoneme respectively.

[3] The key adaptation is in using a bigram model over both graphemes and phonemes, rather than only phonemes as in the original algorithm.

Table 1: Concatenative algorithm for building a subword lexicon that minimizes description length. The input is $\mathcal{T}$, the set of alignments, and a threshold integer k, which is tuned using a held-out development set.

1	Update $pr(g, p)$ by computing the *posterior probabilities* of the tuple (g, p) in $\mathcal{T}$, using the forward-backward algorithm. Repeat once more to get an intermediate lexicon.
2	Compute the Viterbi parse of each $t \in \mathcal{T}$ under the lexicon derived in step 1.
3	Let A, the set of candidate tuples for addition to the lexicon, contain all tuples $(w_i w_{i+1}, r_i r_{i+1})$ such that (w_i, r_i) and (w_{i+1}, r_{i+1}) are adjacent more than k times in the computed Viterbi parses. For each $(g, p) \in A$, estimate the change in description length of the lexicon if (g, p) is added. If description length decreases, remove any null symbols within g and p, and add (g, p) to the lexicon.
4	Repeat steps 1 and 2.
5	Delete all tuples that do not occur in any of the Viterbi parses.
6	Compare the description length of the new lexicon with the lexicon at the start of the iteration. If the difference is sufficiently small, return the new lexicon; else, repeat from step 1.

sequences) in several cases. For example, the bigram `L UW K + S` is much more likely than `L UW K + Z`, since `S` is more likely than `Z` to follow `K`.

To introduce a bias towards phonotacticaly likely bigrams, we define the smoothed bigram probability of the subword a following a subword b. Given that b is made up of a sequence of l phonemes $b_1 b_2 \ldots b_l$, the probability is defined as the interpolation[4]:

$$pr_{new}(a|b) = \lambda_1 pr(a|b_1 b_2 \ldots b_l) + \lambda_2 pr(a|b_1 b_2 \ldots b_{l-1}) + \lambda_3 pr(a|b_1 b_2 \ldots b_{l-2})$$

Both the grapheme and phoneme subword bigrams are smoothed as described.

5 Results

We test our algorithm on the CMU Pronouncing Dictionary[5]. The dictionary is divided randomly into a training (90% of the data) and a test set. Performance is evaluated by measuring the *phoneme error rate* (PER) and the *word error rate* (WER).

The subword extraction algorithm converges in 3 iterations.We run the g2p decoder using the lexicon after 3 iterations, as well as after $0, 1$ and 2 iterations. The results are shown in Table 2.

Figure 1 compares the results of our method (denoted by **'MDL-Sub'**) to two baselines, at different values of *maximum subword length*. To evaluate the quality of our subwords, we substitute another extraction algorithm to create the lexicon – the grow-diag-final phrase extraction method (Koehn et al., 2005), denoted by 'GD' in the figure. We also run the implementation of Bisani and Ney (2008) – denoted by 'BN' – on the same data. BN is an example of a joint-sequence n-gram model, which uses a joint distribution $pr(\mathcal{G}, \mathcal{P})$ of graphemes and phonemes ('graphones'), conditioned on the preceding n-1 graphones for context information. Since this algorithm outperforms most of the existing g2p algorithms, it serves as a good point of comparison to the state of the art in g2p. The results of BN using an n-gram model are compared to MDL-Sub with an n-1 maximum subword length[6].

Table 2: Results after each iteration of subword extraction. While the maximum subword length after iteration 3 is 8, the vast majority of subwords have length 6 or less.

	# subwords	Max subword length	WER	PER
0	$\lvert\mathcal{G}\rvert : 27, \lvert\mathcal{P}\rvert : 40$	1	73.16	24.20
1	$\lvert\mathcal{G}\rvert : 819, \lvert\mathcal{P}\rvert : 1254$	2	48.39	12.43
2	$\lvert\mathcal{G}\rvert : 5430, \lvert\mathcal{P}\rvert : 4954$	4	28.32	7.16
3	$\lvert\mathcal{G}\rvert : 6417, \lvert\mathcal{P}\rvert : 5358$	6	26.31	6.29

The MDL-Sub lexicon does significantly better than the phrases extracted by GD. While BN starts off doing better than MDL-Sub, the latter outperforms BN at longer subword lengths. Most of the additional errors in BN at that stage involve grapheme-to-phoneme ambiguity – phonemes like `AE`, `AA`, and `AH` being confused for one another when mapping

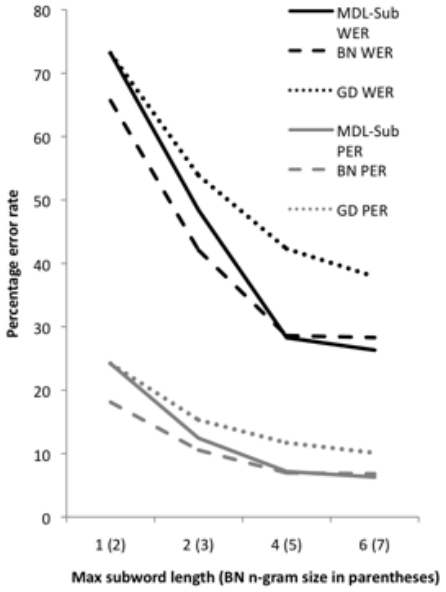
[4]In our experiments, we set $\lambda_1 = 0.5$, $\lambda_2 = 0.3$, $\lambda_3 = 0.2$.

[5]The CMU Pronouncing Dictionary. Available online at http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[6]The contextual information of (n-1)-length subwords with bigrams is assumed to be roughly comparable to that of very short subwords over n-grams.

the grapheme ‘a’, and so on. Far fewer of these errors are produced by our algorithm. However, some of the longer subwords in MDL-Sub do introduce additional errors, mainly because the extraction algorithm merges smaller subwords from previous iterations. For example, one of the items in the extracted lexicon is ‘icati’ – a product of merging ‘ic’ and ‘ati’ – corresponding to `IH K EY SH`, thus generating incorrect pronunciations for words containing the string ‘icating’.

Figure 1: Comparison of error rates.



6 Conclusion

This paper deals with translational ambiguity, which is a major issue in grapheme-to-phoneme conversion. The core of our system consists of extracting subwords of graphemes and phonemes from the training data, so that the ambiguity of deriving a phoneme subword from a grapheme subword is minimized. This is achieved by formalizing ambiguity in terms of the minimum description length principle, and using an algorithm that reduces the description length of the subword lexicon at each iteration.

In addition, we also introduce a smoothing mechanism which retains some of the phonotactic dependencies that may be lost when using subwords rather than singleton letters and phonemes.

While retaining the basic approach to minimizing ambiguity, there are some avenues for improvement. The algorithm that builds the lexicon creates a more or less hierarchical structure – subwords tend to be composed from those extracted at the previous iteration. This appears to be the cause of many of the errors produced by our method. A subword extraction algorithm that does not use a strictly bottom-up process may create a more robust lexicon.

Our method of subword extraction could also be applied to phrase extraction for machine translation, or in finding subwords for related problems like transliteration. It may also be useful in deriving subword units for speech recognition.

References

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50:434–451.

Stanley F Chen. 2003. Conditional and joint models for grapheme-to-phoneme conversion. In *Proceedings of Eurospeech*.

Carl G de Marcken. 1996. *Unsupervised Language Acquisistion*. Ph.D. thesis, MIT.

Sabine Deligne and Frederic Bimbot. 1995. Language modeling by variable length sequences: theoretical formulation and evaluation of multigrams. In *Proceedings of ICASSP*.

Sittichai Jiampojamarn and Grzegorz Kondrak. 2009. Online discriminative training for grapheme-to-phoneme conversion. In *Proceedings of Interspeech*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *Proceedings of IWSLT*.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP*.

Jorma Rissanen. 1978. Modeling by the shortest data description. *Automatica*.

Paul Vozilla, Jeff Adams, Yuliya Lobacheva, and Ryan Thomas. 2003. Grapheme to phoneme conversion and dictionary verification using graphonemes. In *Proceedings of Eurospeech*.

Robert Wagner and Michael Fischer. 1974. The string-to-string correction problem. *Journal of the ACM*.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proceedings of HLT-NAACL*.