Conditional and Joint Models for Grapheme-to-Phoneme Conversion

Stanley F. Chen

IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598
`stanchen@us.ibm.com`

Abstract

In this work, we introduce several models for grapheme-to-phoneme conversion: a conditional maximum entropy model, a joint maximum entropy n-gram model, and a joint maximum entropy n-gram model with syllabification. We examine the relative merits of conditional and joint models for this task, and find that joint models have many advantages. We show that the performance of our best model, the joint n-gram model, compares favorably with the best results for English grapheme-to-phoneme conversion reported in the literature, sometimes by a wide margin. In the latter part of this paper, we consider the task of merging pronunciation lexicons expressed in different phone sets. We show that models for grapheme-to-phoneme conversion can be adapted effectively to this task.

1. Introduction

Grapheme-to-phoneme conversion is the task of converting a word from its spelling (***e.g.***, `wharf`) to its pronunciation or *baseform* (***e.g.***, `W OW R F`); it has application in many areas, most notably speech synthesis and speech recognition. The problem can be framed as follows: given a letter sequence L, find the phone sequence P^* that maximizes $\Pr(P|L)$:

$$P^* = \arg\max_P \Pr(P|L) = \arg\max_P \Pr(L, P) \qquad (1)$$

Thus, one can choose to estimate the *conditional* distribution $\Pr(P|L)$ or, alternatively, the *joint* distribution $\Pr(L, P)$. While there has been extensive work on grapheme-to-phoneme conversion within both of these frameworks, it is still unclear whether one of these approaches is superior to the other.

In this work, we attempt to shed some light on the relative benefits of each paradigm. First, we introduce several new models for grapheme-to-phoneme conversion, including a conditional maximum entropy (ME) model; a joint ME n-gram model; and a joint ME n-gram model that uses syllabification. We demonstrate that these models achieve state-of-the-art performance over several English data sets, sometimes surpassing accuracies reported in the literature by a wide margin. We find that our joint models perform best, and we propose explanations for why this is the case.

In the second part of this paper, we examine the task of merging lexicons that are encoded using different phone sets. We show that performing a context-independent mapping between phone sets yields a surprisingly high error rate, and that models for grapheme-to-phoneme conversion can be adapted successfully to phoneme-to-phoneme conversion. Due to space limitations, many salient details have been omitted in this document; a full description of this work can be found in [1].

2. Grapheme-to-phoneme models

2.1. A conditional maximum entropy model

The most popular models for grapheme-to-phoneme conversion are conditional models expressed as decision trees (***e.g.***, [2, 3]). In this method, there is a hidden *alignment* A describing which phones align with each letter; ***e.g.***, an alignment for *wharf* is

l_i:	*w*	*h*	*a*	*r*	*f*
p_i:	`W`	ϵ	`OW`	`R`	`F`

Typically, the probability $\Pr(P|L)$ is approximated as

$$\Pr(P|A, L) \approx \prod_{i=1}^{m} p(p_i | p_{i-k}^{i-1}, l_{i-k}^{i+k}) \qquad (2)$$

for some appropriate A, where m is the number of letters in L and k is some window size (***e.g.***, 3). The distribution $p(p_i | p_{i-k}^{i-1}, l_{i-k}^{i+k})$ can then be estimated using a decision tree.

In this work, we estimate the distribution $p(p_i | p_{i-k}^{i-1}, l_{i-k}^{i+k})$ using an exponential (or *maximum entropy*) model [4]:

$$p(p_i | x = (p_{i-k}^{i-1}, l_{i-k}^{i+k})) = \frac{\exp(\sum_i \lambda_i f_i(x, p_i))}{\sum_{p'} \exp(\sum_i \lambda_i f_i(x, p'))} \qquad (3)$$

As in typical decision tree work, we consider features $f_i(\cdot)$ that can ask about the identities of phones and letters in specific positions and can form conjunctions of these questions. We use iterative scaling to train the λ_i to optimize the conditional likelihood of the phone sequences given the letter sequences in the training data, modulo a Gaussian prior on the λ_i for regularization [5]. To select which features to include in the model, we use a variation of the gain-based feature induction algorithm described in [4]. Training consists of alternating stages of feature induction and training the λ_i's to convergence.

2.2. A joint maximum entropy n-gram model

Many joint models have been proposed for grapheme-to-phoneme conversion (***e.g.***, [6, 7]). In these models, one has a vocabulary of "chunks" c_i, each consisting of some number of letters paired with some number of phones. Here, we take the chunk set to be any letter by itself, any phone by itself, and any single letter paired with any single phone. Then, we have

$$\Pr(L, P) = \sum_{\substack{C\,:\,\text{letters}(C)\,=\,L,\\ \text{phones}(C)\,=\,P}} \Pr(C) \qquad (4)$$

$$\Pr(C = c_1 \cdots c_m) = \prod_{i=1}^{m} \Pr(c_i | c_1 \cdots c_{i-1}) \qquad (5)$$

We estimate $\Pr(c_i | c_1 \cdots c_{i-1})$ using an ME n-gram model smoothed with a Gaussian prior, as this type of n-gram model has been shown to perform well [5].

To train this model, we considered both the conventional and Viterbi versions of the Expectation-Maximization (EM) algorithm. In the conventional version, we use dynamic programming (DP) to calculate the relevant expectations, and use iterative scaling in the maximization step to update the parameters of the ME n-gram model. Viterbi EM is similar, except that the expectation phase consists of taking counts on the most likely chunking of the training data given the current model, as calculated via DP. We found an effective training schedule to be as follows: We start by training a unigram model with conventional EM. Thereafter, we use Viterbi training, repeatedly increasing the maximum length of n-grams by one; adding features to the model for all n-grams that occur in the Viterbi chunking of the training data; and training to convergence.

2.3. Adding syllabification

An analysis of errors found in runs with the model described in the last section revealed that syllabification information may improve its performance. For example, the model generated the baseform `HH IH L ER Z` (missing a `T`) for the word *Hitler's*. This may be because the model "drew a parallel" with the "*t*" in a word like *hither*, where no `T` phone is present. However, if the system "knew" a syllable boundary followed the "*t*", it would be clear that an analogy with the word *hit* would be more apt.

To achieve this behavior, we modify the model described in Section 2.2 by adding a single chunk to the chunk vocabulary: a chunk representing a syllable boundary. When training on data annotated with syllable boundaries (in both the letter and phone sequences), training is identical to before. When using training data without syllable boundaries, we first insert syllable boundaries using a model trained on supervised data, and then we proceed as before. The most likely placement of syllables given a model can be found via dynamic programming.

3. Grapheme-to-phoneme results

For the experiments in this section, we used the Pronlex lexicon for English (version 0.2), which includes annotation describing the type of each word (*e.g.*, name or acronym). We discarded all entries marked as acronyms or abbreviations, stripped stress markings, and deterministically mapped the phone set down to 41 phones, leaving 98,216 baseforms. From the words with exactly one pronunciation, we randomly selected 2000 words not marked as special, 2000 words marked as (non-foreign) names, and 800 words marked as foreign (400 name and 400 non-name) to be the test set; similarly, a development set of half this size was created, leaving 91,016 baseforms in the training set.

To generate the alignment of the training data needed by the conditional ME model, we first trained a joint n-gram model and computed the Viterbi chunking of the training data with this model. We then derived the desired alignment from this chunking using a process that we do not have the space to describe. To add syllabification to the Pronlex data in order to train syllabified n-gram models, we first built a syllabified n-gram model on a Random House lexicon for which we had segmentation. We used this Random House model to syllabify the Pronlex data.

We tuned various parameters on the development set. For the conditional model, we tuned the window size in eq. (2) ($k = 5$ was best), a single global variance for the Gaussian prior, and when to terminate training. For the joint n-gram models, we tuned the maximum n-gram length and a single global variance parameter. Beam pruning was used in all dynamic programming computations for efficiency; beams were tuned to be as small as possible without affecting performance significantly.

	regular		name		foreign	
	PER	WER	PER	WER	PER	WER
dcs. tree	5.0	27.9	13.5	53.3	24.3	75.4
cond. ME	2.6	14.4	8.9	36.9	19.2	62.6
9-gr.+syl.	1.6	8.3	8.1	32.6	18.0	60.5
8-gr. ME	1.6	8.7	8.0	31.9	17.5	59.3
5-gr. ME	2.0	11.0	8.6	34.0	18.6	61.3

Table 1: Performance of various models on Pronlex; last three rows are joint ME n-gram models (*syl.* = syllabification).

		past result		ME 7-gram	
source	corpus	PER	WER	PER	WER
Pagel [8]	OALD*		21.9		18.9
Marchand [9]	NetTalk		34.5		32.1
Jiang [2]	NetTalk		34.2		34.6
	CMU		26.9		21.2
Galescu [6]	CMU	7.0	28.5	5.9	24.7
	CMU*	9.6	37.4	8.3	32.4
Bisani [7]	Celex	4.0		2.7	

Table 2: Comparison of performance of joint n-gram model against results in literature; "*" signals use of stressed vowels.

As a baseline, we use the publically-available Perl implementation of the conditional decision tree model described by Black *et al.* [3]. We present results on the test set in Table 1; the *regular* column corresponds to non-foreign non-names, *name* is non-foreign names, and *foreign* is foreign words. Phone-error rate (PER) is calculated analogously to word-error rate in speech recognition; word-error rate (WER) is how often the pronunciation of a word is not completely correct.

First, we notice that all of our new models significantly outperform the decision tree baseline. The difference is especially large for "regular" words, with a reduction of over 65% relative in both PER and WER for the joint models. This may be because for regular words, the training set will often contain words very similar to words in the test set; *e.g.*, it may contain the same word with a different suffix. For these cases, models that can memorize long n-grams, like the joint models, will do very well. For the other test sets, the gains are smaller but still large, being over 40% relative and 20% relative in both PER and WER for names and foreign words, respectively.

We see that the conditional ME model outperforms the baseline decision tree substantially, even though both models are conditional and use the same types of questions. This is partially because the baseline model conditions on a smaller window (± 3 rather than ± 5 letters) and does not condition on preceding phones (see eq. (2)), though it may also be partially because the ME model is more effective at combining the same information. Furthermore, we find that our joint models outperform the conditional models tested. We discuss reasons for this in Section 7. For the plain ME n-gram model, we found $n = 8$ to be best on the development set; however, most of the gain can be achieved with $n = 5$, as seen in the last line of Table 1.

While the syllabified n-gram model gave no gain over the plain n-gram, this model can be used to syllabify words. While the Pronlex lexicon does not contain syllable boundaries, we acquired a version of the Random House dictionary that does have this information. This lexicon contained 92,770 words after cleaning, of which we held out 5,000 words to form a test set. Training a syllabified n-gram model on this data, the Viterbi decoding of the test set (given both spellings and baseforms) yielded a precision of 97.4% and recall of 96.8% for predicting the location of syllable boundaries in the baseforms. In contrast, we ran the `tsylb2` rule-based syllabifier [10] (which uses baseforms only) on the test set and achieved a precision of 73.8% and recall of 73.9%.

4. Related work

In this section, we compare the performance of our best model, the joint ME n-gram model, against the best reported results for various English data sets. We attempted to reconstruct the training and test sets used in past work as closely as possible given their descriptions. As most past work does not make use of a separate development set for parameter tuning, we arbitrarily chose beforehand to run all of our contrast runs using an ME 7-gram model with parameter settings optimized for Pronlex. Our contrast results are summarized in Table 2. Some PER results from the original papers have been omitted for the cases where they calculate PER differently than we do. Note that multiple entries for the same corpus may differ in how the corresponding training and test sets were selected; please refer to the original papers for details about each data set.

The ME n-gram achieves performance that is uniformly as good as or better than the best results in the literature, despite the simplicity of this technique. In contrast, Pagel *et al.* [8] use a decision tree given part-of-speech information; Marchand and Damper [9] use *pronunciation by analogy* augmented with 5-way score combination; and Jiang *et al.* [2] use multiple smoothed decision trees with phonemic trigram rescoring.

Galescu and Allen [6] and Bisani and Ney [7] both use methods that can be categorized as joint chunk n-gram models, as described in Section 2.2. From the training data, Galescu and Allen pre-derive a chunk vocabulary consisting of chunks containing at least one letter and one phone; Bisani and Ney incrementally create k:l letter-to-phone chunks for $1 \leq k, l \leq 6$ during training. Our superior performance can most likely be attributed to three factors: First, we use a trivial chunk vocabulary that includes 0:1 and 1:0 chunks; n-grams can model context-dependent phenomena so intelligent chunk induction is unnecessary. Second, we use an excellent smoothing technique that allows us to profitably build 8-gram models and larger; Galescu and Allen note that 5-gram models with Witten-Bell discounting are worse than the corresponding 4-gram models. Finally, we repeatedly realign the training set during training, unlike past work which trains the n-gram models on a fixed chunking.

5. Algorithms for lexicon merging

When merging lexicons that use different phone sets, we assume that the goal is to map each lexicon into the phone set of one particular lexicon, so the task is really one of phone set mapping. A natural thing to try is to manually construct a context-independent mapping between phone sets; however, we have found that this works remarkably poorly. For example, when manually mapping from the Random House dictionary to the Pronlex lexicon, we achieved a PER of 9.0% on a test set composed of "regular" words found in both lexicons. Using the smart context-dependent techniques to be described below, we can achieve a PER of 1.5% on the same test set.

To map a phone sequence in one lexicon to the corresponding sequence in another phone set, we can apply grapheme-to-phoneme algorithms to phoneme-to-phoneme conversion. We can create a training set by selecting words that occur in both lexicons with exactly one pronunciation. As the joint ME n-gram model performed best for grapheme-to-phoneme, we also use this model for phone set mapping.

	regular		name		foreign	
	PER	WER	PER	WER	PER	WER
manual	13.6	55.3	10.6	44.3	23.2	74.0
w/o lett.	3.1	14.7	8.4	34.4	19.8	64.8
w/ lett.	3.2	15.9	6.9	28.9	18.6	62.3

Table 3: Performance of various methods for mapping Orator pronunciations to the Pronlex phone set.

	regular		name	
lexicon	PER	WER	PER	WER
CMUdict	1.4	8.4	5.8	25.1
Random House	1.5	8.1	4.9	19.3
internal IBM	1.3	7.0	3.8	16.4

Table 4: Quality of mapped pronunciations when mapping various lexicons to the Pronlex phone set.

However, sometimes, mapping from one phone set to another cannot be done unambiguously without also looking at the spelling of a word. For example, a phone set used at IBM contains the phone `DX`, which maps to either the phone `D` or `T` in Pronlex depending on whether the corresponding letter in the word is "*d*" or "*t*". While it is possible to extend the joint n-gram to include this extra source of information, we were skeptical that it would work well. Instead, it is straightforward to extend the conditional ME model to also condition on letter information. In the equations in Section 2.1, we simply replace $x = (p_{i-k}^{i-1}, l_{i-k}^{i+k})$ with $x = (p_{i-k}^{i-1}, q_{i-k}^{i+k}, l_{i-k}^{i+k})$ where $q_1 \cdots q_m$ is the phone sequence in the source phone set. For training, we need a 3-way alignment between the letter sequence and both phone sequences. This can be produced by training a grapheme-to-phoneme model and a pure phoneme-to-phoneme model; computing a grapheme-to-phoneme alignment and phoneme-to-phoneme alignment; and merging the two alignments.

6. Results for phone set mapping

For our first set of mapping experiments, instead of mapping from one lexicon to another, we map the output of the commercial Orator system, a rule-based system with dictionary lookup optimized for names [11]. To create a training set for our phone set mapping models, we generated Orator pronunciations for each word with a unique pronunciation in the Pronlex training set described in Section 3. We manually constructed a context-independent phone set map; ran the joint ME n-gram for mapping *without* letter information; and ran the conditional ME model for mapping *with* letter information. To evaluate the quality of the mapping, we compared the mapped pronunciations against the actual Pronlex baseforms for words in our test sets. The results are given in Table 3.

As we can see, the ME models substantially outperform the manual mapping, and on average, the conditional model with letter information slightly outperforms the joint model without letter information. Comparing against the results in Table 1, we see that our best mapped Orator pronunciations actually outperform our best grapheme-to-phoneme model on the names test set. However, note that the mapping models have sufficient

	French		German		Italian	
train set	PER	WER	PER	WER	PER	WER
Pronlex	28.5	69.7	8.7	31.8	15.1	58.4
+RH	21.6	57.9	7.4	27.9	13.2	54.7

Table 5: Performance of merging Random House lexicon with Pronlex on foreign word test sets.

expressive power to correct errors; indeed, as the conditional model has access to the spellings of words, it need not look at the source phone sequence at all to do well. Thus, we cannot separate the contributions of the quality of the original lexicon and the power of the mapping model to the final performance.

In these next experiments, we mapped several lexicons to the Pronlex phone set: CMUdict version 0.6d, a version of the Random House dictionary, and an internal IBM lexicon. We took all words with exactly one baseform in both the source dictionary and Pronlex and extracted 2000 regular words and 1000 names to be a test set; the remaining words formed the training set. (Thus, the test sets were different for each source dictionary.) Results with the conditional ME mapping model on the test sets are presented in Table 4. While the test sets are not identical, comparing these values with those in Table 1 and 3 indicate that using automatically mapped pronunciations from existing lexicons (when available) is superior to using purely automatically generated pronunciations; this would not be the case if using a context-independent phone set mapping.

6.1. Lexicon merging

In our final experiments, we examine whether a grapheme-to-phoneme model trained on the Pronlex lexicon merged with the Random House lexicon can outperform a model trained on Pronlex alone. In particular, the Random House lexicon contains more foreign words with "native" pronunciations (*i.e.*, words pronounced as they would be in their country of origin), and thus should improve performance for these cases. To merge lexicons, we mapped the Random House lexicon to the Pronlex phone set using our conditional ME algorithm and only added pronunciations for words that were not already in Pronlex.

However, since Pronlex does not contain many native pronunciations, it is unlikely that adding Random House baseforms would help much on Pronlex test sets (especially considering phone mapping error), and this is indeed borne out by runs on the Pronlex test. Thus, we created new test sets containing words with native pronunciations; we selected appropriate French, German, and Italian words from an internal IBM English lexicon by using an automatic language identification algorithm and by comparing pronunciations with those found in French, German, and Italian lexicons. This yielded a total of 1510 words; results on this test set are given in Table 5. We see a consistent gain when adding Random House data, demonstrating the effectiveness of lexicon merging when training models.

7. Discussion

In this paper, we introduce several novel methods for grapheme-to-phoneme conversion and show that they significantly outperform existing methods. We propose both joint and conditional models, and find that our joint models yield the best performance. As hypothesized in Section 3, good performance on grapheme-to-phoneme (especially for regular words) may depend largely on an algorithm's ability to memorize long letter/phone sequences in the training data, and n-gram models naturally do this. Decision trees and conditional ME models are much less predisposed to memorizing sequences, and it is not clear how to adapt them to this purpose without sabotaging their strengths.

In addition, joint models do not require pre-aligned data for training, and are a natural way for producing the alignments needed to bootstrap conditional models. Furthermore, joint models are typically symmetric and hence can be used straightforwardly for both grapheme-to-phoneme and phoneme-to-grapheme conversion [6].

However, conditional models have advantages in terms of search. We varied the decoding beam used with a conditional model and a joint model and found that the minimum beam without search errors was $\sim 10^{0.5}$ and $\sim 10^{4}$, respectively. That is, conditional models require a much narrower search, as conditional models predict the next phone conditioning on many letters in the future, while joint models do not have any lookahead at all. As a result, decoding with conditional models takes about half the time as with joint models with our implementations. Finally, we show that conditional models can be effective when mapping a lexicon from one phone set to another, as it is straightforward to condition on multiple information streams in this framework.

8. References

[1] S. F. Chen and B. Maison, "A study of automatic grapheme-to-phoneme conversion," IBM Research, Tech. Rep., 2003, in preparation.

[2] L. Jiang, H.-W. Hon, and X. Huang, "Improvements on a trainable letter-to-sound converter," in *Proc. of Eurospeech*, Rhodes, Greece, 1997.

[3] A. W. Black, K. Lenzo, and V. Pagel, "Issues in building general letter to sound rules," in *Proc. of the ESCA Workshop on Speech Synthesis*, Australia, 1998, pp. 77–80.

[4] A. Berger, S. Della Pietra, and V. Della Pietra, "A maximum entropy approach to natural language processing," *Computational Ling.*, vol. 22, no. 1, pp. 39–71, 1996.

[5] S. F. Chen and R. Rosenfeld, "A survey of smoothing techniques for maximum entropy models," *IEEE Trans. on Speech and Audio Proc.*, vol. 8, no. 1, pp. 37–50, 2000.

[6] L. Galescu and J. F. Allen, "Bi-directional conversion between graphemes and phonemes using a joint n-gram model," in *Proc. of the 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, Scotland, 2001.

[7] M. Bisani and H. Ney, "Investigations on joint-multigram models for grapheme-to-phoneme conversion," in *Proc. of ICSLP*, 2002.

[8] V. Pagel, K. Lenzo, and A. W. Black, "Letter to sound rules for accented lexicon compression," in *Proc. of ICSLP*, Sydney, Australia, 1998.

[9] Y. Marchand and R. I. Damper, "A multi-strategy approach to improving pronunciation by analogy," *Computational Ling.*, vol. 26, no. 2, pp. 195–219, 2000.

[10] W. Fisher, "A C implementation of Daniel Kahn's theory of English syllable structure," `ftp://jaguar.ncsl.nist.gov/pub/tsylb2-1.1.tar.Z`, 1996.

[11] M. Macchi and M. Spiegel, "Using a demisyllable inventory to synthesize names," *Speech Technology*, pp. 208–212, 1990.