PROBABILISTIC DEDUCTION OF SYMBOL MAPPINGS FOR EXTENSION OF LEXICONS

Rita Singh[1], *Evandro B. Gouvêa*[1] *and Bhiksha Raj*[2]

[1]Carnegie Mellon University, Pittsburgh, PA, USA
[2]Mitsubishi Electric Research Labs, Cambridge, MA, USA

Abstract

This paper proposes a statistical mapping-based technique for guessing pronunciations of novel words from their spellings. The technique is based on the automatic determination and utilization of unidirectional mappings between n-tuples of characters and n-tuples of phonemes, and may be viewed as a statistical extension of analogy-based pronunciation guessing algorithms.

1. Introduction

A commonly encountered problem in the deployment of speech-based user interfaces is that of *grapheme to phoneme* (G2P) conversion, i.e., guessing the pronunciations of novel words from their spellings. This is an important problem both for text-to-speech conversion systems that must guess the correct pronunciations of words in order to synthesize them, and speech-to-text systems that must know how a word is pronounced in order to be able to recognize it.

Arguably the most informed approach to determining the pronunciation of a word is from acoustic realizations of it, e.g. [1]. Given a number of recorded utterances of the word and a number of candidate pronunciations for it, the most likely of these pronunciations is determined using a speech recognizer. However, acoustic realizations are often not available and the pronunciations must be guessed from the orthography – the spellings of the words alone. Even when acoustic realizations are available, spelling-to-pronunciation conversion mechanisms may be required to generate the candidate pronunciations to be evaluated.

A large number of techniques have been proposed in the literature for the determination of pronunciations from spellings. The simplest approach derives the pronunciation of a word from (parts of) other words with known pronunciation that are spelt analogously to the target word [2]. Other techniques derive pronunciations from expert-enumerated rules that relate them to spellings, e.g. [3, 4]. Decision-tree based G2P conversion systems attempt to derive such rules automatically from pronunciation dictionaries that are presented as training data [5]. Some authors have also attempted to formalize the inference task performed by the decision trees through statistical models that explicitly represent the branching rules within the trees as functions in an exponential probability model [6]. Neural network based G2P methods represent the relationship between spellings of words and their pronunciations through a neural network, whose parameters are learnt from training data [7].

Statistical pronunciation guessing algorithms typically attempt to capture the sequential dependencies in spellings and pronunciations through appropriate statistical models. HMM-based techniques have long been particularly popular. These methods usually represent the orthography for a word as the output of an HMM, whose underlying state sequence represents the phoneme sequence for the word [8]. Lexical dependencies between the phonemes themselves may be formalized through the manner in which the HMMs for phonemes are linked together [9].

Joint multi-gram techniques such as [10, 11] model the joint distribution of spellings and pronunciations with an N-gram model. Each spelling-pronunciation pair is assumed to consist of a sequence of unit pairs, where each unit pair includes one symbol unit from the spelling and one from the pronunciation. The symbol units themselves do not merely comprise individual characters or phonemes, but rather *sequences* of them. Since the number of characters in a spelling are often different from the number of phonemes in the pronunciation, alignments between the spelling and pronunciation are estimated such that each character/phoneme maps onto a sequence of one or more phonemes/characters. A unit pair thus consists of either one character and an associated group of phonemes or *vice versa*. The probabilities of unit pairs are represented through an N-gram model. Since spellings and pronunciations are treated symmetrically, the N-gram model may be used either to derive a pronunciation given a spelling or *vice versa*.

In this paper we present a statistical symbol-mapping approach to G2P conversion that is based on the automatic determination and utilization of mappings between sequences (or n-tuples) of characters in the spelling and sequences (or n-tuples) of phonemes in the pronunciation. Mechanically, the proposed technique may hence be viewed as a statistical extension of analogy-based mapping [2]. However, the technique may also be viewed as a generalization of HMM-based G2P methods such as [8], since sequences of phoneme n-tuples are assumed to be generated by a Markovian process, and these in turn are assumed to generate spelling n-tuples. As in HMM-based methods, all parameters of this generative process are learned from a training pronunciation dictionary. The technique may also be related to joint multi-gram methods, the chief difference being that we model n-tuples, rather than N-grams. Also, the proposed method is not bi-directional; rather one symbol sequence is assumed to generate the other. Unlike multi-gram methods that depend on back-off mechanisms to control overparametrization, there is no explicit back-off in our scheme. Rather, back-off is implicit in that mappings are learnt for n-tuples of all orders (up to a predetermined n). Also, unlike multi-gram methods, at no point is it necessary to explicitly compute alignments between spellings and pronunciations – mappings are learned through an EM algorithm that considers all possible mappings.

Experimental evaluation using the CMU dict shows that the proposed method is able to learn mappings effectively and gen-

erate pronunciations of higher accuracy than two standard pronunciation guessing algorithms. Resulting pronunciation dictionaries are also found to result in speech recognition accuracies that are better than those obtained with the other two techniques.

The rest of the paper is arranged as follows. In Section 2 we describe our statistical model and the learning algorithm to obtain its parameters. In Section 3 we describe how the learned parameters may be used to obtain the pronunciation for a novel word. In Section 4 we report our experiments, and finally in Section 5 we present our conclusions.

2. Probabilistic Mapping between Symbol Sequences

We treat the phoneme set as a set of symbols or an *alphabet*. Similarly, the characters used in the orthography of words represent a second alphabet. Dictionary words and their pronunciations are assumed to be the output of a common generator which produces output symbol sequences from the two different alphabets. We aim to find the underlying rules that relate one output sequence to the other. We assume that the underlying rules exist, are probabilistic and can be described in terms of probabilities of *mappings* between sequences of symbols from the two alphabets.

Consider two symbol sets $\{\mathrm{X}\}$ and $\{\mathrm{Y}\}$ respectively. They may be composed of different alphabets and may be of different sizes. Consider the case where a common input or *generator* independently generates output symbol strings from each set. Let these output symbol strings be denoted by $x_i x_j x_k ...$ and $y_i y_j y_k ...$ respectively. These may be of different lengths. We define any sequence of n symbols to be an *n-tuple*. We define a *substring* to be a segment of a complete output string. We assume, based on the presence of the common generator, that there exist *mappings* between substrings (or n-tuples) of the two output symbol strings of the kind

$$\begin{array}{rcl@{\qquad}rcl} x_i & \mapsto & y_i & x_i & \mapsto & y_i y_j \\ x_i & \mapsto & y_i y_j y_k & x_i x_j & \mapsto & y_i \\ x_i x_j & \mapsto & y_i y_j & x_i x_j & \mapsto & y_i y_j y_k \\ x_i x_j x_k & \mapsto & y_i & x_i x_j x_k & \mapsto & y_i y_j \\ x_i x_j x_k & \mapsto & y_i y_j y_k & & \cdots & \end{array}$$

We define a *mapping order* of N_1,N_2 as one where mappings are defined for substrings of up to N_1 symbols from $\{\mathrm{X}\}$ and substrings of up to N_2 symbols from $\{\mathrm{Y}\}$. The symbol $\mapsto$ denotes an onto mapping. We make the following assumptions about the mappings:

1. The mappings are of a definite order N_1,N_2, that is known. In practice, this order must be assumed.
2. Forward and reverse mappings are distinct. Thus, for a mapping of order N_1, N_2, the forward mappings map n-tuples of order up to N_1 from $\{\mathrm{X}\}$ on to n-tuples of order up to N_2 from $\{\mathrm{Y}\}$, while the reverse mappings map n-tuples of order up to N_2 from $\{\mathrm{Y}\}$ on to n-tuples of order up to N_1 from $\{\mathrm{X}\}$. The mappings are directed, i.e., the mapping $x_i x_j \mapsto y_i y_j y_k$ is not identical to the mapping $y_i y_j y_k \mapsto x_i x_j$.
3. Mappings are not reducible, i.e. if $x_1 x_2 x_3 \mapsto y_1 y_2$ and $x_1 x_2 \mapsto y_1$, it does not imply that $x_3 \mapsto y_2$.
4. Both alphabets include a *begin* and an *end* symbol that identify the beginning and ending of every symbol sequence. The *begin* symbol from one alphabet by itself only maps on to the *begin* symbol from the other alphabet. Similarly, by themselves the two *end* symbols only map on to one another. Substrings from one alphabet that include the *begin* symbol may only map onto substrings from the other alphabet that also include the *begin* symbol. Likewise for the *end* symbol.
5. The mappings are probabilistic, *i.e.*,

$$\begin{array}{lll} y_i\ y_j\ ... \mapsto & x_i \text{ with P}(y_i\ y_j\ ... \mapsto x_i) & \forall \text{ i,j} \\ & x_i\ x_j \text{ with P}(y_i\ y_j\ ... \mapsto x_i\ x_j) & \forall \text{ i,j} \\ & \vdots & \end{array}$$

$P(x_1 x_2 x_3 \mapsto y_1 y_2)$ need not be equal to $P(y_1 y_2 \mapsto x_1 x_2 x_3)$.

The various mapping probabilities may now be computed from a set of instances of parallel symbol sequences from both alphabets. In order to deduce the mappings, we begin by creating a symbol set $\{\mathrm{H}\}$ consisting of *all* n-tuples up to order N_1 from $\{\mathrm{X}\}$, and a symbol set $\{\mathrm{K}\}$ consisting of all n-tuples up to order N_2 from $\{\mathrm{Y}\}$:

$$\begin{array}{l} h_i \in \{\mathrm{H}\} = \text{the sequence } x_i..x_j \text{ of length } n : n \leq N_1\ \forall\ i, j \\ k_i \in \{\mathrm{K}\} = \text{the sequence } y_i..y_j \text{ of length } n : n \leq N_2\ \forall\ i, j \end{array}$$

Note that for any $N_1 > 1$, there are multiple sequences of n-tuples from $\{\mathrm{H}\}$ that map onto a given symbol sequence $x_1 x_2 \ldots$ We assume that the *a priori* probability of an n-tuple sequence $h_1 h_2 \ldots$ is given by the Markovian model $P(h_1)P(h_2|h_1)P(h_3|h_2)\ldots$ The terms to be estimated are now the probabilities of the mappings $P(h_i \mapsto k_j)$ and the transition probabilities $P(h_i|h_j)$.

Given a symbol sequence $\mathcal{X}$ from $\{\mathrm{X}\}$ and a corresponding symbol sequence $\mathcal{Y}$ from $\{\mathrm{Y}\}$, we first identify the set of all n-tuples h_i in $\mathcal{X}$, and all n-tuples k_j in $\mathcal{Y}$. The set of valid n-tuple sequences is restricted not only by the fact that they must compose $\mathcal{X}$, but also by the fact that they must map onto a corresponding n-tuple sequence that composes $\mathcal{Y}$. This restriction can be represented by a graph as illustrated by Figure 2.

Each path θ in the graph represents the event that a specific n-tuple sequence $h_a h_b h_c \ldots$ maps onto a specific n-tuple sequence $k_a k_b k_c \ldots$ of the same length. The likelihood of the path θ is given by

$$P(\theta) = \begin{array}{l} P(h_a \mapsto k_a)P(h_b \mapsto k_b)\ldots P(h_n \mapsto k_n) \\ P(h_a)P(h_b|h_a)\ldots P(h_n|h_m) \end{array} \quad (1)$$

The total likelihood that the symbol sequence $\mathcal{X}$ maps onto the sequence $\mathcal{Y}$ is the sum of the likelihood of all paths through the graph:

$$P(\mathcal{X} \mapsto \mathcal{Y}) = \sum_{\theta} P(\theta) \quad (2)$$

.

Each node in the graph corresponds to the mapping of a specific n-tuple h_i (on the X-axis) onto an n-tuple k_i (on the Y axis). The probability that h_i maps onto k_j in the process of mapping $\mathcal{X}$ onto the $\mathcal{Y}$, $P(h_i, k_j, \mathcal{X}, \mathcal{Y})$ is the sum of all the likelihood of all paths in the graph that pass through the node(s) that map h_i onto k_j and can be computed using a forward-backward algorithm. Similarly, $P(h_j, h_i, \mathcal{X}, \mathcal{Y})$, the total probability of all paths that step through the n-tuple h_j immediately after the n-tuple h_i can also be computed from the forward-backward algorithm.

The process of estimating mapping and transition probabilities is now simple: given a set of symbol sequences $\{\mathcal{X}\}$ from

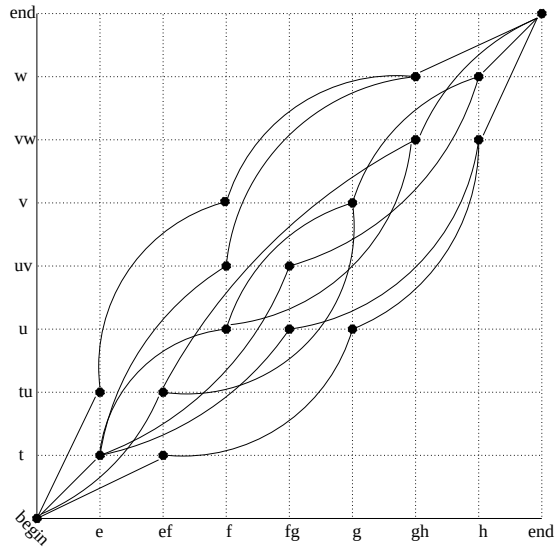

Figure 1: Valid mapping paths for the strings **tuvw** $\mapsto$ **efgh**, on a graph. The vertical columns refer to N_1-tuples constructed from the alphabet in {X}. Each of these is an element of {H}. The horizontal column consists of N_2-tuples constructed from the alphabet in {Y}. Each of these is an element of the set {K}.

{X} and a corresponding set of sequences $\{\mathcal{Y}\}$ from {Y}, we obtain the following iterative update rules for the mapping and transition probabilities through the EM algorithm:

$$P^{n+1}(k_j|h_i) = \frac{\sum_{\mathcal{X}\in\{\mathcal{X}\}}\sum_{\mathcal{Y}\in\{\mathcal{Y}\}} P^n(k_j,h_i,\mathcal{X},\mathcal{Y})}{\sum_{k'_j}\sum_{\mathcal{X}\in\{\mathcal{X}\}}\sum_{\mathcal{Y}\in\{\mathcal{Y}\}} P^n(k'_j,h_i,\mathcal{X},\mathcal{Y})} \quad (3)$$

$$P^{n+1}(h_j|h_i) = \frac{\sum_{\mathcal{X}\in\{\mathcal{X}\}}\sum_{\mathcal{Y}\in\{\mathcal{Y}\}} P^n(h_j,h_i,\mathcal{X},\mathcal{Y})}{\sum_{h'_j}\sum_{\mathcal{X}\in\{\mathcal{X}\}}\sum_{\mathcal{Y}\in\{\mathcal{Y}\}} P^n(h'_j,h_i,\mathcal{X},\mathcal{Y})} \quad (4)$$

where the superscript $n+1$ refers to estimates obtained in the $n+1^{\text{th}}$ iteration of the algorithm, and the superscript n represents probabilities computed from the n^{th} iteration of transition and mapping probability estimates. Equations (3) and (4) are iterated until the combined probabilities of all graphs constructed from all $\mathcal{X},\mathcal{Y}$ pairs in the training set converge.

3. Generating sequences from $\{Y\}$ given sequences from $\{X\}$

Once the parameters of the statistical model relating {X} to {Y} are learnt they can be used to estimate the unknown sequence $\hat{\mathcal{Y}}$ of symbols from {Y} that best corresponds to a given sequence $\mathcal{X}$ of symbols from {X}. We estimate $\hat{\mathcal{Y}}$ as the most likely sequence for the given $\mathcal{X}$.

$$\hat{\mathcal{Y}} = \text{argmax}_{\mathcal{Y}} P(\mathcal{Y}|\mathcal{X}) \quad (5)$$

For any given $\mathcal{Y}$, $P(\mathcal{Y}|\mathcal{X})$ can be computed from the graph that represents the mapping of the component n-tuples of $\mathcal{X}$ onto the n-tuples of $\mathcal{Y}$ using Equation 2. This would however be an extremely expensive calculation, since the entire forward-backward procedure must be followed for every $\mathcal{Y}$ in order to identify the most likely one.

To simply the procedure, instead we jointly identify the most likely n-tuple sequences $\mathcal{H}$ (that composes $\mathcal{X}$ and $\mathcal{K}$ (derived from {K}, the set of all possible n-tuples from the symbols in {X}), given $\mathcal{X}$.

$$\hat{\mathcal{K}} = \text{argmax}_{\mathcal{H},\mathcal{K}} P(\mathcal{H}\mapsto\mathcal{K}|\mathcal{X}) \quad (6)$$

This can be computed very simply using the Viterbi algorithm over a graph such as the one in Figure 3. The X-axis on this graph comprises all n-tuples that can be obtained from $\mathcal{X}$. The Y-axis comprises all n-tuples in {K}. Any valid path through the graph must represent a valid sequence of n-tuples along the X-axis (i.e. that represents a tokenization of $\mathcal{X}$). $\hat{\mathcal{K}}$ is the n-tuple sequence corresponding to the most likely path through this graph. The symbol sequence $\hat{\mathcal{Y}}$ underlying $\hat{\mathcal{K}}$ becomes our best guess for $\mathcal{Y}$.

We note here that since the procedure is ML, the best estimate for the n-tuple k associated with an n-tuple h is always the one that has the highest $P(k|h)$ value, i.e. the mapping of any h to a k now becomes deterministic. Therefore, the problem of identifying the most likely n-tuple sequence $\mathcal{K}$ simply reduces to that of finding the best segmentation of the given $\mathcal{X}$ into an n-tuple sequence. However, in the Viterbi procedure the $P(k|h)$ terms do have an effect on the outcome: n-tuples for which $\max_k P(k|h)$ values are low are de-weighted with respect to n-tuples with high $\max_k P(k|h)$ values. In other words, the algorithm preferentially selects n-tuple segmentations where the mappings of the n-tuples are more certain.

The G2P problem can now be simply stated in terms of the above mathematics: the spellings of words are assumed to represent sequences from the character set {X} and their pronunciations the corresponding sequences from the phoneme set {Y}. Mappings of character n-tuples to phoneme n-tuples are learnt using the procedure outlined in Section 2 from a training dictionary. Thereafter, the pronunciations of new words are derived from their spelling using the procedure in Section 3.

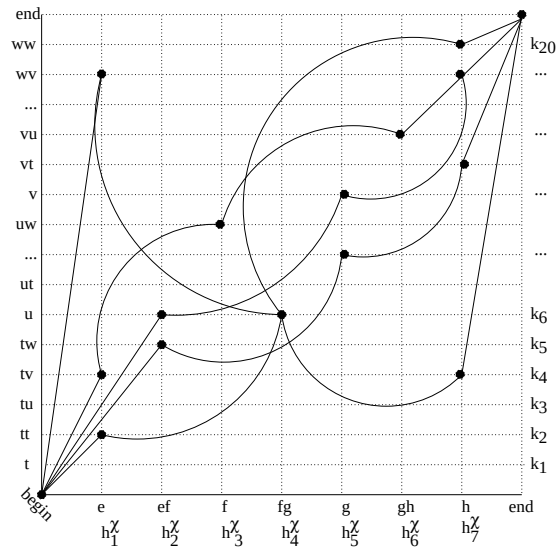

Figure 2: A few paths on a graph used for estimating an unknown $\mathcal{Y}$ given a string $\mathcal{X}$ = **efgh**. $\mathcal{Y}$ could in principle be composed of any n-tuple sequence from {K}. In this example the elements of {K} are all the n-tuples up to N_2 that can possibly be composed from of the alphabet of {Y}, which is specified here to be { **t,u,v,w**}. The abscissa has only those symbols from {H} which correspond to $\mathcal{X}$. The ordinate has all symbols from {K}. The mapping orders N_1 and N_2 are assumed to be 2.

4. Experimental results

We conducted two experiments to compare the proposed G2P algorithm. The CMU dictionary has 130,000 entries represented by a 50-symbol phone set. We learned order 4,3 map-

Table 1: Phoneme error in guessed pronunciations.

Algorithm	ADDTTP	LTS	Symbol Map
Phone error (%)	22.5	28.5	18.8

Table 2: Recognition error with manually-crafted and automatically generated dictionaries.

Algorithm	Manual	ADDTTP	LTS	Sym. Map
Word err. (%)	9.7	30.7	35.9	24.8

ping probabilities from 70,000 randomly selected entries. We then generated the pronunciations for the remaining 60,000 entries using the algorithm of Section 3. As a comparison, we also computed pronunciations using two other tools, the NIST addttp package that utilizes a combination of N-gram probabilities and decision-tree like rules to generate pronunciations [12] and the CMU LTS decision-tree-based pronunciation guessing tool that is included as part of the CMU SphinxTrain toolkit [13]. The NIST addttp package is trained on word pronunciations obtained from a variety of sources including the CMUdict. We trained the LTS package on the same subset of the CMUdict that our mapping probabilities were also trained on. We report the *phone error* obtained with each of the three techniques on our test set in table 1. The phone error represents the percentage of phonemes in the manually tagged pronunciations of the words in the test set that were correctly captured by the G2P algorithm. It is clear from the table that the proposed symbol map technique clearly outperforms the other two methods. CMUdict has been noted to have several erroneous or inconsistent pronunciations. The results in table 1 compare the pronunciations obtained with the G2P algorithms against the pronunciation given in the dictionary. It is entirely possible that in at least a few cases the pronunciation returned by the algorithms is, in fact, *better* than the one in the dictionary, although a direct comparison between the two would label it as an error. In order to eliminate this possibility, we also ran a recognition test. In this test we attempted to recognize proper names. The test data were obtained from www.talkhouse.com and consisted of 1075 recordings from over 30 speakers uttering combinations of various song titles, artist names, etc. Not unusually for song titles and artist names, several of the words were oddly spelt, potentially making them difficult to guess pronunciations for. None of the names were in the CMUdict. Four separate dictionaries were used in the experiment – a hand-crafted dictionary, and dictionaries produced by ADDTTP, LTS and the proposed symbol-mapping algorithm. Both LTS and the mapping probabilities for the symbol-mapping algorithm were learnt from the entire CMUdict. The acoustic models were fully continuous HMMs trained from other data also obtained from Talkhouse. A bigram LM was trained from a large collection of song titles, album names etc., including those in the test data. Table 2 shows the results obtained with the four dictionaries. Not unexpectedly (given the difficulty of guessing the pronunciations of song titles and artist names), by far the best recognition accuracies are still obtained with the manually crafted dictionary. Of the automatically obtained dictionaries, the one obtained with the symbol-mapping algorithm results in the best recognition performance, while the other two are significantly worse.

5. Discussion and Conclusion

As a result of the MAP approach taken in Section 3, the symbol mapping algorithm is primarily a statistical extension of analogy based G2P algorithms. As a result, the pronunciations generated can sometimes be egregious, for instance the pronunciation for the word "ABC" was guessed to be "AE B K", while both the addttp and lts algorithms correctly identified it as "EY B IY S IY". A large reason for this is that the algorithm in its current form does not utilize constraints on phoneme sequences. Better performance may be obtained by inverting the mapping procedure, describing the spelling as the output of a Markovian chain of pronunciations. The current implementation also relies heavily on implicit back-off – if some of the potential n-tuples in a sequence have zero probability due to not having been observed during training, the sequence is described in terms of shorter n-tuples for which probabilities are known. Better performance may be obtained by smoothing all transition probabilities and applying explicit back-off policies. Finally, the current implementation expects the order of mapping to be strictly left to right – a later n-tuple from $\mathcal{X}$ will perforce map onto a later n-tuple from $\mathcal{Y}$. This is not mandatory – by permitting arbitrary transitions over the n-tuple components of $\mathcal{Y}$, we can permit corresponding n-tuples of $\mathcal{X}$ and $\mathcal{Y}$ to be differently arranged, such as may be encountered when translating between words in languages. These issues will be explored in future work.

6. References

[1] G. K. Anumanchipalli, M. Ravishankar, and R. Reddy, "Acoustic-based improving pronunciation inference using n-best list, acoustics and orthography," in *Proc. ICASSP*, 2007.

[2] R. Damper, Y. Marchand, M. Adamson, and K. Gustafson, "Comparative evaluation of letter-to-sound conversion techniques for English text-to-speech synthesis," in *Proc. SSW-3*, 1998, pp. 53–58.

[3] H. M. Meng, S. Seneff, and V. W. Zue, "The use of higher level linguistic knowledge for spelling-to-pronunciation generation," in *Proc. ISSIPNN*, vol. 2, 1994, pp. 670–673.

[4] M. Ravishankar and M. Eskenazi, "Automatic generation of context-dependent pronunciations," in *Proc. Eurospeech*, vol. 5, 1997, pp. 2467–2470.

[5] A. W. Black, K. Lenzo, and V. Pagel, "Issues in building general letter to sound rules," in *Proc. SSW-3*, 1998, pp. 77–80.

[6] S. F. Chen, "Conditional and joint models for grapheme-to-phoneme conversion," in *Proc. Eurospeech*, 2003, pp. 2033–2036.

[7] N. Deshmukh, J. Ngan, J. Hamaker, and J. Picone, "An advanced system to generate pronunciations of proper nouns," in *Proc. ICASSP*, vol. 1, 1997, pp. 1467–1470.

[8] F. Jelinek, *Statistical Methods for Speech Recognition*. MIT Press, 1997.

[9] P. Taylor, "Hidden markov models for grapheme to phoneme conversion," in *Proc. Eurospeech*, 2005, pp. 1973–1976.

[10] L. Galescu and J. F. Allen, "Bi-directional conversion between graphemes and phonemes using a joint n-gram model," in *Proc. 4th ITRW on Speech Synthesis*, 2001.

[11] M. Bisani and H. Ney, "Investigations on joint-multigram models for grapheme-to-phoneme conversion," in *Proc. ICSLP*, vol. 1, 2002, pp. 105–108.

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

[13] [Online]. Available: http://cmusphinx.org/tutorial.html