UNSUPERVISED, LANGUAGE–INDEPENDENT GRAPHEME–TO–PHONEME CONVERSION BY LATENT ANALOGY

Jerome R. Bellegarda

Spoken Language Group
Apple Computer, Inc.
Cupertino, California 95014

ABSTRACT

Automatic, data-driven grapheme-to-phoneme conversion is a challenging but often necessary task. The top-down strategy implicitly followed by traditional inductive learning techniques tends to dismiss relevant contexts when they have been seen too infrequently in the training data. The bottom-up philosophy inherent in, e.g., pronunciation by analogy, allows for a markedly better handling of rarer contexts, but proves nonetheless equally dependent on local, language-dependent alignments between letters and phonemes. This paper proposes an alternative bottom-up approach, dubbed *pronunciation by latent analogy*, which adopts a global definition of analogy, more amenable to obviate such supervision. For each out-of-vocabulary word, a neighborhood of globally relevant pronunciations is constructed through an appropriate data-driven mapping of its graphemic form. Phoneme transcription then proceeds via locally optimal sequence alignment and maximum likelihood position scoring. This method was successfully applied to the speech synthesis of proper names with a large diversity of origin.

1. INTRODUCTION

Assigning phonemic/phonetic transcriptions to graphemic word forms is of critical importance to all spoken language applications. In automatic speech recognition (ASR), the search module relies on phonetic transcriptions to select appropriate acoustic models against which to score the input utterance. Likewise, in text-to-speech (TTS) synthesis, phonemic transcriptions are required for the selection of the proper TTS units from which to generate the desired waveform. Depending on the particular emphasis and/or application considered, the process of constructing such transcriptions goes by different names, e.g., grapheme-to-phoneme conversion [1], baseform generation [2], pronunciation modeling [3], letter-to-sound translation [4], and text-to-phoneme mapping [5], to name but a few. Since this paper is primarily motivated by a speech synthesis application, we adopt the terminology "grapheme-to-phoneme conversion" (GPC).

For most languages, especially English, GPC is a challenging task. One approach is to rely on expert knowledge from trained linguists to manually create each entry in an underlying pronunciation dictionary. But this is time-consuming, inherently language-dependent, often prone to inconsistencies, and not applicable to the real-time processing of words not present in the dictionary (out-of-vocabulary words). This has sparked interest in data-driven GPC methods, which leverage a variety of statistical algorithms to automatically derive pronunciation from orthography. In ASR applications, this is typically done by exploiting the available acoustic side information (see, e.g., [3]). In TTS applications, existing pronunciation dictionaries serve as training data to extract suitable conversion rules, whose role is to encapsulate language regularity within a manageably small number of general principles (cf. [4]).

Such extraction is usually based on inductive learning techniques involving, e.g., decision trees [6] or Bayesian networks [7]. Take the case of a decision tree. The problem is to classify an input grapheme sequence into the appropriate output phoneme sequence. Training therefore proceeds using sequence pairs, aligned with the help of language-dependent "allowables,", i.e., individual pairs of letters and phonemes which are allowed to correspond [8]. The tree is first grown by iteratively splitting nodes to minimize some measure of spread (e.g., entropy), and then pruned back to avoid unnecessary complexity and/or overfitting. During classification, the tree is traversed on the basis of questions asked about the context of each grapheme, until a leaf corresponding to a particular phoneme (or phoneme string) is reached. As is well known, question selection and pruning strategy heavily influence classification granularity and generalization ability. Typical tree designs attempt to strike an acceptable balance between the two.

But this trade-off has a price: the effective set of questions ends up best covering those phenomena which are most represented in the training data. In contrast, rarely seen contexts tend to be overlooked, regardless of their degree of similarity or relevance in a given situation. For out-of-vocabulary words which largely conform to the general principles derived from the training sequences, this is relatively inconsequential. But many other words, such as names (especially those of foreign origin), may comprise a number of irregular patterns rarely seen in the dictionary, for which this limitation may be more deleterious.

Hence the interest in bottom-up, rather than top-down, strategies for GPC, e.g., pronunciation by analogy [9]. The idea is to assemble the pronunciation for an unknown word by matching substrings of the input to substrings of known lexical words, hypothesizing a partial pronunciation for each matched substring derived from local, individual letter-phoneme alignments, and concatenating the partial pronunciations. This strategy results in markedly better handling of rarer contexts [10], but the letter-phoneme alignment process on which it relies remains problematic, due to the need for "allowables" as mentioned above. Although recent work has attempted to relax some of this supervision, the handling of "nulls" would seem to remain highly language-dependent [11].

The goal of this paper is to present an alternative bottom-up approach, more amenable to unsupervised processing, in which we decouple the two sub-problems of finding neighbors and assembling the pronunciation. When finding neighbors, the concept of analogy is cast in a global (latent) sense, which circumvents the need for individual letter-phoneme alignments. Then, when assembling the pronunciation, local information emerges automatically from the influence of the entire neighborhood, which bypasses the need for any external linguistic knowledge. The paper

is organized as follows. The next section gives a general overview. Sections 3 and 4 examine the two main building blocks of the proposed approach. Finally, Section 5 reports the outcome of preliminary experiments conducted on a diverse corpus of proper names.

2. OVERVIEW

While traditional inductive learning techniques tend to exhibit good performance on "conforming" words, they degrade rapidly when encountering patterns unusual for the language considered [8]. As an illustration, consider the proper name of Indian origin: *Krishnamoorthy*, for which the phoneme sequence:

k r I S n @ m **U** r T i (1)

represents a good estimate of the correct pronunciation. Because this name is not part of the underlying Apple dictionary, the GPC converter currently bundled with MacOS X comes up with the (incorrect) sequence:

k r I S n { m **u** 3 r D i (2)

after traversing a dedicated 2000-node decision tree trained on 56K names (predominantly of Western European origin).

Comparing (2) with (1), three errors stand out: (i) the schwa "@" is replaced by the full vowel "{", (ii) the unvoiced "T" is replaced by the voiced version "D", and (iii) the stressed vowel "**U**" is replaced by the improper compound "**u** 3". These errors can all be traced to poor generalization properties. Specifically, the ending "3 r D i" results from the influence of a large number of names in the training dictionary ending in "orthy," such as *Foxworthy*. The vowel compound comes from the inability of this pattern to account for "oo," hence the awkward attempt to have it both ways by concatenating the two vowels. Finally, the full vowel "{", commonly seen after "n" in names like *McNamara*, points to an obvious failure to connect *Krishnamoorthy* with the more closely related *Krishna*.

This example underscores the importance of exploiting *all potentially relevant contexts*, regardless of how sparsely seen they may have been in the training data. In order to do so, pronunciation by analogy eschews the inherently top-down stategy of decision trees in favor of a bottom-up approach relying on the concept of lexical neighborhood. Loosely speaking, two words are lexical neighbors if they share a common graphemic substring [9]. The problem with this concept is that it offers no principled way of deciding which neighbor(s) of a new word can be deemed to substantially influence its pronunciation. As a case in point, from a Levenshtein string-edit distance perspective, the words *although*, *through*, and *enough* would likely be considered lexical neighbors, while in fact they have absolutely no bearing on each other when it comes to the pronunciation of the substring "ough."

What seems to be needed is a concept of neighborhood which is not specified exclusively in terms of graphemic substrings, but also includes phonemic information. This leads us to the following notion of *orthographic neighborhood*.

Given an out-of-vocabulary word, we define its orthographic neighborhood as the set of in-vocabulary words which are sufficiently "close" to it, in some suitable metric adopted from latent semantic analysis (LSA). LSA has already proven effective over the past decade in a variety of other fields, including query-based information retrieval, word clustering, document/topic clustering, large vocabulary language modeling, and semantic inference for voice command and control [12]. Here, LSA is used to determine what grapheme strings are most characteristic of words, and map all in-vocabulary words onto the space of all characteristic grapheme strings. The outcome is a set of *orthographic anchors* (one for each in-vocabulary word), determined automatically from the underlying vocabulary. Each out-of-vocabulary word is then compared to each orthographic anchor, and the corresponding "closeness" evaluated. If this closeness is high enough, the associated in-vocabulary word is added to the orthographic neighborhood of the out-of-vocabulary word.

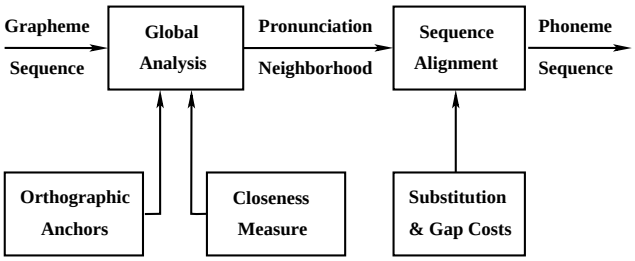

Fig. 1. *Pronunciation by Latent Analogy.*

Neighborhood entries can thus be thought of as the orthographic generalization of candidate pronunciation variants in ASR, and the LSA formalism is seen to play a role comparable to that of contextual questions in the decision tree framework. Compared to the latter, however, it offers several benefits. First, it obviates the need for explicit, language-dependent alignments between individual pairs of letters and phonemes. Second, grapheme substrings with hidden similarities having a bearing on pronunciation are automatically discovered and exploited. And third, closeness is defined globally across an entire sequence, so words with higher relevance across multiple contexts can easily be identified.

Once an orthographic neighborhood is available for a given out-of-vocabulary word, it is straightforward to gather the corresponding set of pronunciations from the existing dictionary. We refer to this set as the *pronunciation neighborhood*. Phonetic expansions in the pronunciation neighborhood have the property to contain at least one sub-string which is "locally close" to the pronunciation sought. The next step is therefore to automatically align these pronunciations to find common elements between them. The more common a particular element in a particular position, the more likely it is to belong to the out-of-vocabulary phonetic transcription. Thus, the maximum likelihood estimate at every position is the best candidate for the final pronunciation. The procedure, illustrated in Fig. 1, is entirely data-driven and requires no human supervision.

3. ORTHOGRAPHIC NEIGHBORHOODS

Let $\mathcal{V}$, $|\mathcal{V}| = M$, be a collection of words of interest (e.g., a set of names), and $\mathcal{T}$, $|\mathcal{T}| = N$, the set of all strings of n letters[1] that can be produced from this vocabulary (including markers for word beginning and ending). Typically, M varies between 10,000 and 100,000; with the choice $n = 3$, N is of the order of 10,000.

We first construct a $(N \times M)$ matrix W, whose entries w_{ij} suitably reflect the extent to which each n-letter string $t_i \in \mathcal{T}$ ap-

[1] Of course, strings of n graphemes could also be used, but this does not seem to have an appreciable effect on performance. Note that opting for letters rather than graphemes preserves language independence.

peared in each word $w_j \in \mathcal{V}$. From [12], a reasonable expression for w_{ij} is:

$$w_{i,j} = (1 - \varepsilon_i) \, \frac{c_{i,j}}{n_j} \, , \tag{3}$$

where $c_{i,j}$ is the number of times t_i occurs in word w_j, n_j is the total number of n-letter strings present in this word, and ε_i is the normalized entropy of t_i in $\mathcal{V}$. The global weighting implied by $1 - \varepsilon_i$ reflects the fact that two n-letter strings appearing with the same count in a particular word do not necessarily convey the same amount of information; this is subordinated to the distribution of the n-letter strings in the entire vocabulary $\mathcal{V}$.

Then we perform a singular value decomposition (SVD) of W [12] as:

$$W = U \, S \, V^{\,T} \, , \tag{4}$$

where U is the $(N \times R)$ left singular matrix with row vectors u_i $(1 \leq i \leq N)$, S is the $(R \times R)$ diagonal matrix of singular values $s_1 \geq s_2 \geq \ldots \geq s_R > 0$, V is the $(M \times R)$ right singular matrix with row vectors v_j $(1 \leq j \leq M)$, $R \ll M, N$ is the order of the decomposition, and ^{T} denotes matrix transposition.

This (rank-R) decomposition defines a mapping between: (i) the set of n-letter strings in $\mathcal{T}$ and, after appropriate scaling by the singular values, the R-dimensional vectors $\bar{u}_i = u_i S$ $(1 \leq i \leq N)$, and (ii) the set of words in $\mathcal{V}$ and, again after appropriate scaling by the singular values, the R-dimensional vectors $\bar{v}_j = v_j S$ $(1 \leq j \leq M)$. The latter are the orthographic anchors mentioned in the previous section. The dimension R is bounded from above by the rank of the matrix W, and from below by the amount of distortion tolerable in the decomposition. Values of R in the range $R = 50$ to $R = 100$ seem to work well.

By definition, the R-dimensional vector space spanned by the vectors $\bar{u}_i$ and $\bar{v}_j$ minimally describes the linear space spanned by W, i.e., the underlying vocabulary $\mathcal{V}$ and set of n-letter strings $\mathcal{T}$. Thus, the relative positions of the orthographic anchors in that space reflect a parsimonious encoding of the orthography used in the training data. This means that any out-of-vocabulary word mapped onto a vector "close" (in some suitable metric) to a particular orthographic anchor would be expected to be closely related to the corresponding in-vocabulary word, and conversely any in-vocabulary word whose orthographic anchor is "close" to a particular vector in the space would tend to be related to the corresponding out-of-vocabulary word. This offers a basis for determining orthographic neighborhoods.

To proceed, however, we first have to specify how to represent an out-of-vocabulary word, say $\tilde{w}_p$ (where $p > M$), in the above vector space. For each n-letter string in $\mathcal{T}$, we compute for this word the weighted counts (3) with $j = p$. The resulting feature vector, a column vector of dimension N, can be thought of as an additional column of the matrix W. Thus, assuming the matrices U and S do not change appreciably, the SVD expansion (4) implies:

$$\tilde{w}_p = U \, S \, \tilde{v}_p^{\,T} \, , \tag{5}$$

where the R-dimensional vector $\tilde{v}_p^{\,T}$ act as an additional column of the matrix $V^{\,T}$. This in turn leads to the definition:

$$\tilde{\bar{v}}_p = \tilde{v}_p \, S = \tilde{w}_p^{\,T} \, U \, . \tag{6}$$

It remains to define an appropriate closeness measure to compare $\tilde{\bar{v}}_p$ to each of the $\bar{v}_j$'s. From [12], a natural metric to consider is the cosine of the angle between them. Thus:

$$K(\tilde{\bar{v}}_p, \bar{v}_j) = \cos(\tilde{v}_p S, v_j S) = \frac{\tilde{v}_p \, S^{\,2} \, v_j^{\,T}}{\|\tilde{v}_p S\| \, \|v_j S\|} \, , \tag{7}$$

for any $1 \leq j \leq M$. Using (7), it is a simple matter to rank all orthographic anchors in decreasing order of closeness to the representation of a given out-of-vocabulary word. The associated orthographic neighborhood follows by retaining only those entries whose closeness measure is higher than a pre-set threshold.

4. SEQUENCE ALIGNMENT

Such orthographic neighborhood comprises (in-vocabulary) word entries. From the underlying dictionary, we can easily deduce the associated pronunciation neighborhood (comprising phoneme strings). In principle, each phoneme string in this pronunciation neighborhood contains at least one sub-string which is germane to the input out-of-vocabulary word. Thus, the final pronunciation can be assembled by judicious alignment of appropriate phoneme sub-strings from the pronunciation neighborhood.

To carry out such alignment, we adopt a sequence analysis approach commonly used in molecular biology. In bioinformatics, a number of algorithms have been developed to align similar protein sequences in order to find groups of related proteins, and ultimately identify likely genes in the genome. The basic framework is dynamic programming, with provisions for the existence of gaps in the alignment, and the possibility of specific amino acid pairings. To find maximally homologous sub-sequences, techniques have also been devised to locally align specific regions of two protein sequences (see, e.g., [13]). This approach has recently been modified for orthographic sequences to enable computer-assisted morphological lemma discovery [14].

We proceed in analogous fashion to apply the underlying framework to pronunciation alignment. Assume, without loss of generality, that we want to align two phoneme strings $\varphi_1 \ldots \varphi_k \ldots \varphi_K$ and $\psi_1 \ldots \psi_\ell \ldots \psi_L$ (of length K and L, respectively) from the pronunciation neighborhood, and denote by $A(k, \ell)$ the best (minimum cost) alignment between $\varphi_1 \varphi_2 \ldots \varphi_k$ and $\psi_1 \psi_2 \ldots \psi_\ell$. If $C(k, \ell)$ is the cost of substituting phoneme ψ_ℓ for phoneme φ_k, $g(i, k)$ the cost of a gap $\varphi_i \ldots \varphi_k$ in the first string, and $h(j, \ell)$ the cost of a gap $\psi_j \ldots \psi_\ell$ in the second string, the basic dynamic programming recursion can be written:

$$A(k, \ell) = \min\{A(k-1, \ell-1) + C(k, \ell), G(k, \ell), H(k, \ell)\} \, , \tag{8}$$

where:

$$G(k, \ell) \;=\; \min_{0 \leq i \leq k-1} \{A(i, \ell) + g(i, k)\} \, , \tag{9}$$

$$H(k, \ell) \;=\; \min_{0 \leq j \leq \ell-1} \{A(k, j) + h(j, \ell)\} \, , \tag{10}$$

with initial conditions $A(k, 0) = h(0, k)$, $1 \leq k \leq K$ and $A(0, \ell) = g(0, \ell)$, $1 \leq \ell \leq L$.

We select as first reference phoneme sequence the entry corresponding to the closest orthographic anchor comprising a beginning-of-word marker, as determined above. We then proceed in left-to-right fashion until we have aligned the pronunciation of every word in the neighborhood to its immediate predecessor. The maximum likelihood estimate is then computed for every position, by simply using the observed phoneme counts at this position. The outcome is the final pronunciation sought.

5. EXPERIMENTS

Preliminary experiments were conducted using an underlying training vocabulary of $M = 56,514$ names, predominantly of Western

European origin. At this stage, lexical stress markers were omitted from all pronunciations, leaving for future work the important aspect of stress assignment. For this vocabulary the number of unique 3-letter strings turned out to be $N = 8,257$. Also available was a disjoint set of $84,193$ names reflecting a much larger diversity of origin. From this set we extracted as test data a subset of 500 names whose pronunciations comprised 3160 phonemes.

As baseline system, we used the decision tree-based GPC converter bundled with MacOS X, previously mentioned in Section 2. On the above (difficult) test set, the phoneme error rate (taking into account substitutions, insertions, and deletions) was 23.3%, and the pronunciation error rate 80.2%. That is, only 1 in 5 phoneme sequences was actually error-free when compared to a manually derived pronunciation produced by a human expert.

We performed the SVD of the $(8257 \times 56,514)$ matrix W constructed from the training data using the single vector Lanczos method [12]. Orthographic anchors were obtained using $R = 100$ for the order of the decomposition. Each of the (out-of-vocabulary) words in the test data was then compared to these orthographic anchors and the resulting orthographic and pronunciation neighborhoods assembled accordingly, with the thresholds chosen so that on the average each neighborhood comprised about 200 entries.

Entries in each pronunciation neighborhood were then aligned with a rather primitive version of (8), where: (i) exact phoneme matches were encouraged with a zero substitution cost, (ii) vowel-consonant substitutions were prohibited with an infinite substitution cost, and (iii) substituting a vowel (respectively a consonant) for any other vowel (respectively any other consonant) was given the same penalty as introducing a gap—the latter clearly adopting a highly simplistic view of phonology, especially regarding vowels. The final pronunciation was produced using the maximum likelihood estimate at each position, as described earlier.

On the above test set, we observed a phoneme error rate of 13.4%, and a pronunciation error rate of 38.0%. Thus, despite the above simplification, pronunciation by latent analogy shows substantial improvement compared to the decision tree method. To illustrate, for the name *Krishnamoorthy*, the expansion:

$$\text{k r I S n @ m } \mathbf{u} \text{ r T i} \qquad (11)$$

was the pronunciation returned by the proposed approach. A comparison with (1) and (2) shows that, while still not entirely correct, this expansion solves many of the problems observed in decision tree GPC. This bodes well for the general applicability of the method to generic GPC in speech synthesis applications.

6. CONCLUSION

Common inductive learning techniques used in automatic GPC (e.g., decision trees) do not always generalize well, as in the case of proper names of foreign origin for example. In contrast, methods based on pronunciation by analogy exploit all potentially relevant contexts, regardless of how sparsely seen they may have been in the training data. But they tend to require external linguistic knowledge, be it for local letter-phoneme alignments, or the discovery of paradigmatic relationships. We have proposed an alternative bottom-up strategy, completely unsupervised, which decouples the two sub-problems of finding neighbors and assembling the pronunciation.

By redefining the concept of analogy in terms of grapheme-phoneme co-occurrences, it becomes possible to construct neighborhoods of globally relevant pronunciations, using a latent semantic analysis framework operating on n-letter graphemic forms. Phoneme transcription then follows via locally optimal sequence alignment and maximum likelihood position scoring, in which the influence of the entire neighborhood is implicitly and automatically taken into account. This method was observed to be effective on a difficult test corpus of proper names with a large diversity of origin.

These results should be regarded as work-in-progress. We have not yet studied the influence of n on the vector space of characteristic grapheme strings, and the likely trade-off between modeling power and generalization properties should be made explicit. Likewise, we have not yet exploited the full power of the sequence alignment framework, and in particular more realistic substitution and gap costs should be investigated. Finally, our current position scoring strategy could benefit from a variety of refinements, such as the integration of confidence measures in the final estimation. Future efforts will concentrate on addressing these important points.

7. REFERENCES

[1] O. Andersen *et al.*, "Comparison of Two Tree–Structured Approaches for Grapheme–to–Phoneme Conversion," in *Proc. ICSLP*, Philadelphia, PA, pp. 1700–1703, October 1996.

[2] B. Ramabhadran *et al.*, "Acoustics–Only Based Automatic Phonetic Baseform Generation," in *Proc. ICASSP*, Seattle, WA, pp. 309–312, May 1998.

[3] W. Byrne *et al.*, "Pronunciation Modeling Using a Hand–Labelled Corpus for Conversational Speech Recognition," in *Proc. ICASSP*, Seattle, WA, pp. 313–316, May 1998.

[4] V. Pagel *et al.*, "Letter–to–Sound Rules for Accented Lexicon Compression," in *Proc. ICSLP*, Sydney, Australia, pp. 2015–2018, December 1998.

[5] J. Suontausta and J. Häkkinenen, "Decision Tree Based Text–to–Phoneme Mapping for Speech Recognition," in *Proc. ICSLP*, Beijing, China, pp. 831–834, October 2000.

[6] A.K. Kienappel and R. Kneser, "Designing Very Compact Decision Trees for Grapheme–to–Phoneme Transcription," in *Proc. Eurospeech*, Aalborg, Denmark, pp. 11911–1914, September 2001.

[7] C.X. Ma and M.A. Randolph, "An Approach to Automatic Phonetic Baseform Generation Based on Bayesian Networks," in *Proc. Eurospeech*, Aalborg, Denmark, pp. 1453–1456, September 2001.

[8] A.W. Black, K. Lenzo, and V. Pagel, "Issues in Building General Letter-to-Sound Rules," in *Proc. 3rd Int. Workshop Speech Synthesis*, Jenolan Caves, Australia, pp. 77–80, December 1998.

[9] F. Yvon, "Paradigmatic Cascades: a Linguistically Sound Model of Pronunciation by Analogy," in 35th *Ann. Meeting Assoc. Computational Linguistics*, pp. 428–435, 1997.

[10] R.I. Damper *et al.*, "Evaluating the Pronunciation Component of Text-to-Speech Systems for English: A Performance Comparison of Different Approaches," *Computer Speech and Language*, Vol. 13, No. 2, pp. 155–176, 1999.

[11] R.I. Damper *et al.*, "A Pronunciation-by-Analogy Module for the Festival Text-to-Speech Synthesiser," in *Proc. 4th Int. Workshop Speech Synthesis*, Pilochry, Scotland, pp. 97–102, August 2001.

[12] J.R. Bellegarda, "Exploiting Latent Semantic Information in Statistical Language Modeling," *Proc. IEEE*, Vol. 88, No. 8, pp. 1279–1296, August 2000.

[13] M. Vingron, "Near–Optimal Sequence Alignment," *Curr. Op. Struct. Biology*, Vol. 6, No. 3, pp. 346–352, June 1996.

[14] A. Dalli, "Biologically Inspired Lexicon Structuring Technique," in *Proc. Human Lang. Technol. Workshop*, San Diego, CA, pp. 341–343, March 2002.