A Latent Analogy Framework for Grapheme–to–Phoneme Conversion

Jerome R. Bellegarda

Spoken Language Group, Apple Computer, Inc.
Cupertino, California 95014, USA
jerome@apple.com

Abstract

Data-driven grapheme-to-phoneme conversion involves either (top-down) inductive learning or (bottom-up) pronunciation by analogy. As both approaches rely on local context information, they typically require some external linguistic knowledge, e.g., individual grapheme/phoneme correspondences. To avoid such supervision, this paper proposes an alternative solution, dubbed *pronunciation by latent analogy*, which adopts a more global definition of analogous events. 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 synthesis of proper names with a large diversity of origin.

1. Introduction

Data-driven grapheme-to-phoneme conversion (GPC), in which an input grapheme sequence is automatically classified into the appropriate output phoneme sequence, is a challenging but often necessary task. In most text-to-speech 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. Such extraction tends to involve either inductive learning (IL) or pronunciation by analogy (PbA).

IL systems typically use such top-down techniques as decision trees [1] or Bayesian networks [2]. The relevant conversion rules are then effectively embodied in the structure of the associated tree or network. While these systems exhibit good performance on "conforming" words, they tend to degrade rapidly when encountering patterns unusual for the language considered [1]. Given a suitable (string-based) measure of similarity between words, PbA systems directly retrieve partial pronunciations for local fragments of the input word, which are then concatenated to obtain the final pronunciation [3]. This is beneficial in the case of rarer contexts, but complicates the resolution of potential conflicts [4]. Note that both IL and PbA methods are implicitly supervised, since they critically rely on external linguistic knowledge—be it for (language-dependent) correspondences between individual graphemes and phonemes, or the discovery of paradigmatic relationships. More recently, a mixed-strategy approach was also proposed, based on joint grapheme-phoneme n-grams [5]. This technique appears to relax some of these constraints, albeit at the expense of greater data and computational requirements.

This paper proposes an alternative solution, dubbed *pronunciation by latent analogy*. As the name implies, it follows the same bottom-up strategy as PbA, but expands the definition of analogous events to make it more robust and more amenable to unsupervised processing. This is achieved by decoupling the two sub-problems of the classical approach, i.e., finding similar words (neighbors) and assembling the pronunciation. The concept of analogy is cast in a global (latent) sense, which circumvents the need for individual letter-phoneme alignments, while local information emerges automatically from the influence of the entire neighborhood, which bypasses the need for any other external linguistic knowledge.

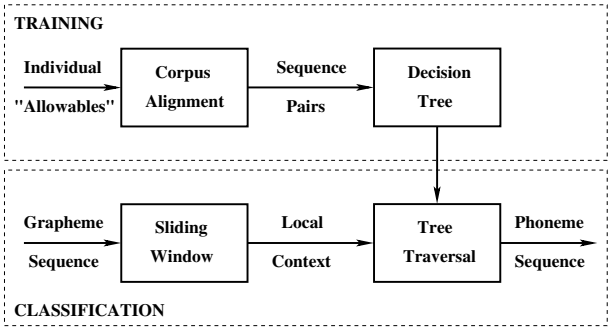

Figure 1: *Typical Top-Down GPC Framework.*

The next section contrasts the proposed approach with existing GPC solutions. In Sections 3 and 4, we examine in greater details the two main building blocks of the resulting framework: orthographic neighborhoods and sequence alignment. Finally, Section 5 reports the outcome of an experimental evaluation conducted on a diverse corpus of proper names.

2. Overview

Fig. 1 illustrates the traditional IL framework in the case of a decision tree. Training 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 [1]. These "allowables" are typically derived by hand, though there is some preliminary evidence that they can also be satisfactorily inferred using a version of the EM algorithm [5]. 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, a sliding window is used to isolate the local context of each grapheme, and the tree is traversed on the basis of questions asked about this context. When the most likely leaf is reached, the decision tree returns the corresponding phoneme (or phoneme string) associated with the local input grapheme.

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

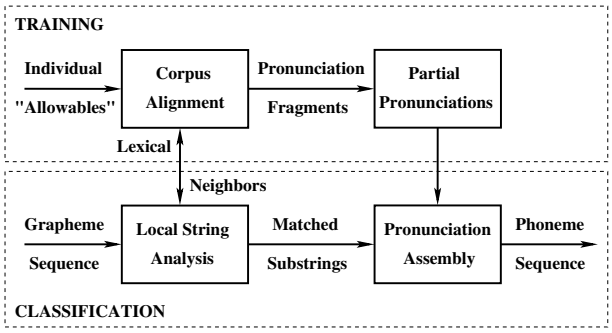

Figure 2: *Typical Bottom-Up GPC Framework.*

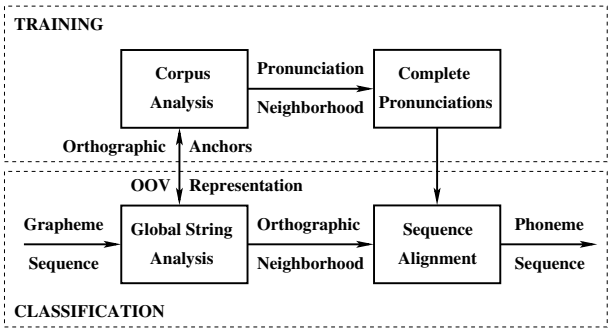

Figure 3: *Pronunciation by Latent Analogy Framework.*

in the training data. In contrast, rarely seen contexts tend to be overlooked, regardless of their degree of relevance to a given situation. For OOV 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.

This underscores the importance of exploiting *all potentially relevant contexts*, regardless of how sparsely seen they may have been in the training data. The associated (PbA) framework is illustrated in Fig. 2. As before, a local analysis is performed on the input grapheme sequence, but this time a (discrete) measure such as the Levenshtein string-edit distance is used to determine promising lexical neighbors. Loosely speaking, two words are lexical neighbors if they share a common graphemic substring [3]. The idea is to match substrings of the input to substrings of its lexical neighbors, and to convert the various matched substrings to partial pronunciations, again with the help of suitable "allowables." The final pronunciation is then assembled by concatenating all the partial pronunciations.

This strategy allows for a better handling of unusual patterns [4], but the letter-phoneme alignment process on which it relies remains problematic. Although recent work has attempted to relax some of this supervision, the handling of "nulls" would seem to remain highly language-dependent [6]. Perhaps even more importantly, this concept 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 distance perspective, the words "*rough*," "*though*," and "*through*" 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 (local) graphemic substrings, but also includes a modicum of phonemic information. Recently, there have been attempts to define a pronunciation model based on joint grapheme-phoneme n-grams [5], [7]. But n-grams tend to involve a large number of parameters, and are unable to take into account long-range dependencies [8]. This paper pursues an alternative avenue: instead of incorporating phonemic information directly into the concept of neighborhood, we postulate that phonemic consistency implicitly correlates with some of the global properties of graphemic substrings. Thus, given an OOV word, we define its *orthographic neighborhood* as the set of in-vocabulary (IV) words which are *globally* "close" to it. For this we rely on a suitable metric adopted from latent semantic analysis (LSA), a paradigm which has already proven effective in a variety of other fields [8]. Here, LSA is used to (i) determine what grapheme strings are most characteristic of words, and (ii) map all IV words onto the space of all characteristic grapheme strings. The outcome is a set of automatically determined *orthographic anchors*, one for each IV word.

Each OOV word is then compared to each orthographic anchor, and the corresponding "closeness" evaluated. If this closeness is high enough, the associated IV word is added to the orthographic neighborhood of the OOV word. The procedure is therefore analogous to classical PbA, except that LSA now defines the concept of similarity in a global rather than a local sense. Once an orthographic neighborhood is available for a given OOV word, it trivially leads to the *pronunciation neighborhood*, i.e., the set of associated pronunciations from the existing dictionary. In contrast with classical PbA, this step does not involve partial but complete pronunciations, since specific local substring information is no longer available.

Phonetic expansions in the pronunciation neighborhood have the property to contain at least one substring 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 OOV phonetic transcription. Thus, the maximum likelihood estimate at every position is the best candidate for the final pronunciation. The procedure, illustrated in Fig. 3, is entirely data-driven and requires no human supervision.

3. Orthographic Neighborhoods

Let $\mathbf{V}$, $|\mathbf{V}| = M$, be a collection of words of interest (e.g., a set of proper names), and $\mathbf{T}$, $|\mathbf{T}| = N$, the set of all strings of n graphemes (or, more generally, letters) that can be produced from this vocabulary (including a marker 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. Following the procedure detailed in [8], we first construct a $(N \times M)$ matrix W, whose entries w_{ij} suitably reflect the extent to which each n-letter string $t_i \in \mathbf{T}$ appeared in each word $w_j \in \mathbf{V}$. We then perform a singular value decomposition (SVD) of W as:

$$W = U\,S\,V^{T}\,, \tag{1}$$

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 $\mathbf{T}$ and, after appropriate scaling

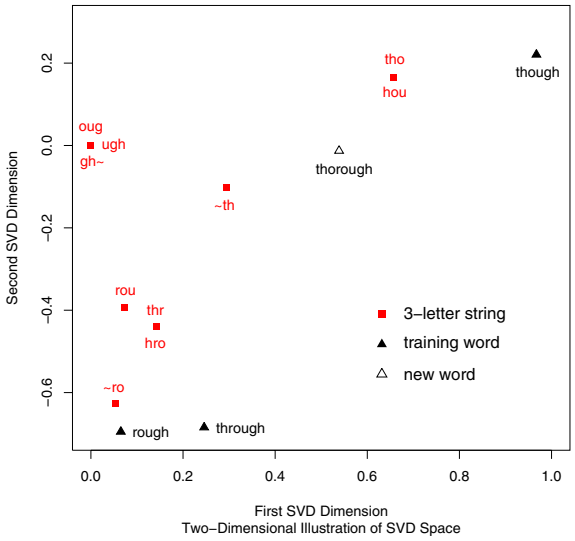

Figure 4: *Typical Behavior in R-Dimensional Space ($R = 2$).*

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 $\mathbf{V}$ and, again after scaling, the R-dimensional vectors $\bar{v}_j = v_j S$ $(1 \leq j \leq M)$. The latter are the orthographic anchors sought. 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 $\mathbf{V}$ and set of n-letter strings $\mathbf{T}$. 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 OOV 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 IV word, and conversely any IV word whose orthographic anchor is "close" to a particular vector in the space would tend to be related to the corresponding OOV word. This offers a basis for determining orthographic neighborhoods.

To illustrate, consider the collection of the $N = 3$ words mentioned previously: (i) "*rough*," (ii) "*though*," and (iii) "*through*." Recall that, although (i) is a proper subset of (iii), and (ii) and (iii) differ by only one letter, the associated pronunciations have little in common: this is one of these (not-so-rare) cases where local graphemic evidence can be misleading. Instead, we will exploit n-letter co-occurrence information. In this simple example, there are only $M = 10$ strings of $n = 3$ letters in the collection, counting those formed with the word boundary marker ˜. This makes for some interesting patterns: for example, "*tho*" and "*hou*" always co-occur, and uniquely identify "*though*," while "*oug*," "*ugh*," and "*gh˜*" appear in all three words. Intuitively, it makes sense that this kind of differentiation would be useful for neighborhood disambiguation.

Constructing the (3×10) co-occurrence matrix and performing the SVD, we obtain the 2-dimensional space depicted in Fig. 4. This figure shows how each word and each 3-letter string is represented in the space. Note that the letter strings which each uniquely identify a word—"*˜ro*" for (i), "*tho/hou*" for (ii), and "*thr/hro*" for (iii)—each score relatively high on one of the main axes. Conversely, the letter string "*oug/ugh/gh˜*," which conveys no information about the identity of a word, is located at the origin. On the other hand, the orthographic anchors for the three words fall "close" to the letter strings which predict them best. And, not surprisingly, letter strings which are present in two different words indeed appear in the vicinity of both—cf. "*rou*" for (i)–(iii), and "*˜th*" for (ii)–(iii). Now consider what happens with a new word, such as "*thorough*," which was not seen during training but is otherwise consistent with the global graphemic structure. As it turns out, the associated representation is indicated by the hollow triangle in Fig. 4. This point is closest to the orthographic anchor associated with "*though*," rather than that associated with "*through*." Thus, despite a closer Levenshtein relationship with (iii), the new word can be considered orthographically most related to (ii). In this case, this means that we are able to automatically infer the correct pronunciation for the substring "*ough*."

To proceed, however, we first have to specify how to represent an OOV word, say $\tilde{w}_p$ (where $p > M$), in the above vector space. For each n-letter string in $\mathbf{T}$, we compute for this word the appropriate entries w_{ij} 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, the SVD expansion (1) implies $\tilde{w}_p = US\tilde{v}_p^T$, which in turn leads to the definition:

$$\tilde{\bar{v}}_p = \tilde{v}_p\, S = \tilde{w}_p^T\, U\,. \tag{2}$$

It remains to define an appropriate closeness measure to compare $\tilde{\bar{v}}_p$ to each of the $\bar{v}_j$'s. From [8], a natural metric to consider is the cosine of the angle between them. Thus:

$$K(\tilde{\bar{v}}_p, \bar{v}_j) = \cos(\tilde{v}_p S, v_j S) = \frac{\tilde{v}_p\, S^2\, v_j^T}{\|\tilde{v}_p S\|\, \|v_j S\|}\,, \tag{3}$$

for any $1 \leq j \leq M$. Using (3), it is a simple matter to rank all orthographic anchors in decreasing order of closeness to the representation of a given OOV 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 (IV) 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 substring which is germane to the input OOV word. Thus, the final pronunciation can be assembled by judicious alignment of appropriate phoneme substrings from the pronunciation neighborhood. The problem is somewhat more complex than in classical PbA, however, because of the lack of readily specified matched grapheme substrings and associated partial phoneme sequences. Here, the entities being aligned are the complete phoneme strings, as opposed to local grapheme and phoneme sub-sequences. In other words, the relevant phoneme substrings have to emerge from the alignment itself.

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., [9]). This approach has recently been modified for orthographic sequences to enable computer-assisted morphological lemma discovery [10].

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

$$A(k,\ell) = \min\{A(k-1,\ell-1) + C(k,\ell), G(k,\ell), H(k,\ell)\}\,, \tag{4}$$

where:

$$G(k,\ell) = \min_{0\le i\le k-1}\{A(i,\ell) + g(i,k)\}\,, \tag{5}$$

$$H(k,\ell) = \min_{0\le j\le \ell-1}\{A(k,j) + h(j,\ell)\}\,, \tag{6}$$

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

We select as first reference phoneme sequence the entry corresponding to the closest orthographic anchor comprising a word boundary marker at the beginning, 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

It is natural to evaluate the performance of GPC algorithms on proper names, since they tend to represent the largest proportion of OOV words (especially in name-intensive applications like email, directory assistance, and news reading). Unfortunately, no standard resource exists on which to run such evaluation, which makes direct comparison between various methods difficult. On available test sets, PbA systems seem to achieve around 15% phoneme error rate and 50% pronunciation error rate, while decision trees and joint n-grams seem to achieve around 10% phoneme error rate and between 30 and 45% pronunciation error rate, depending on the amount of linguistic side information available [7]. These results may in part be influenced by the fairly homogeneous nature of the data. As baseline system, we therefore opted for a dedicated 2000-node decision tree-based system trained on a set of $M = 56,514$ (predominantly Western European) names. Lexical stress markers were omitted from all pronunciations, leaving for future work the important aspect of stress assignment. We then selected as test set a subset of 500 names extracted from a disjoint set of $84,193$ names reflecting a much larger diversity of origin and ethnicity. The associated pronunciations comprised 3160 phonemes. On this (difficult) set, the baseline phoneme error rate was 23.3%, and the pronunciation error rate 80.2%.

For the training vocabulary the number of unique 3-letter strings turned out to be $N = 8,257$. We performed the SVD of the resulting $(8257 \times 56,514)$ matrix W using the single vector Lanczos method [8]. Orthographic anchors were obtained using $R = 100$ for the order of the decomposition. Each of the (OOV) 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 average each neighborhood comprised about 200 entries.

Entries in each pronunciation neighborhood were then aligned with a rather primitive version of (4), 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.

We observed a phoneme error rate of 13.4%, and a pronunciation error rate of 38.0%. Thus, the phoneme and pronunciation error rates improved by approximately 40% and 50%, respectively. This shows the ability of the proposed algorithm to exploit the small amount of evidence present in the training data which proved germane to certain categories of foreign names. Note, however, that the pronunciation error rate remains high on this difficult task, presumably because other categories were not represented at all.

6. Conclusion

We have proposed a bottom-up GPC approach which decouples the two sub-problems of finding neighbors and assembling the pronunciation. By recasting the concept of analogy in terms of global co-occurrences between graphemes and phonemes, 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 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.

7. References

[1] 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, Dec. 1998.

[2] 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, Sep. 2001.

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

[4] 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 Lang.*, Vol. 13, No. 2, pp. 155–176, 1999.

[5] L. Galescu and J. Allen, "Bi-directional Conversion Between Graphemes and Phonemes Using a Joint N-gram Model," in *Proc. 4th Int. Workshop Speech Synth.*, Pitlochry, Scotland, Aug. 2001.

[6] R.I. Damper, C.Z. Stanbridge, and Y. Marchand, "A Pronunciation-by-Analogy Module for the Festival Text-to-Speech Synthesiser," in *Proc. 4th Int. Workshop Speech Synth.*, Pitlochry, Scotland, pp. 97–102, Aug. 2001.

[7] L. Galescu and J. Allen, "Pronunciation of Proper Names with a Joint N-Gram Model for Bi-Directional Conversion Grapheme–to–Phoneme Conversion," in *Proc. ICSLP*, Denver, CO, pp. 109–112, Sep. 2002.

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

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

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