Technical Report CSRG-402
Department of Computer Science
University of Toronto
December 1999

Alignment of Phonetic Sequences

Grzegorz Kondrak[1]

Abstract

Alignment of phonetic sequences is a necessary step in many applications in computational phonology. In this paper, I critically evaluate several approaches to phonetic alignment that have been reported within the last few years. I then present a new algorithm that is geared towards the alignment of cognates. I show how the basic dynamic programming algorithm for sequence comparison can be modified to deal with a range of phonological phenomena. For computing phonetic similarity, I propose a scoring scheme that is based on multivalued features. Finally, I provide a complete test set that demonstrates the new algorithm performs better on cognate alignment, in terms of accuracy and efficiency, than other algorithms reported in the literature.

Keywords: alignment, distance, features, phonetic, sequence

Available at:
`http://www.cs.utoronto.ca/csri/reports.html`

[1]Computational Linguistics Research Group, Department of Computer Science, University of Toronto, 10 King's College Road, Toronto, Ontario, Canada M5S 1A4. e-mail: kondrak@cs.toronto.edu

Contents

1 Introduction

Identification of the corresponding segments in sequences of phones is a necessary step in many applications in both diachronic and synchronic phonology. Usually we are interested in aligning sequences that represent forms that are related in some way: a pair of cognates, or the underlying and the surface forms of a word, or the intended and the actual pronunciations of a word. An example alignment of two cognates, Spanish *poner* and Italian *porre*, is shown in Table 1. Alignment of phonetic sequences presupposes transcription of sounds into discrete phonetic segments, and so differs from matching of utterances in speech recognition. On the other hand, it has much in common with the alignment of proteins and DNA sequences. Many methods developed for molecular biology can be adapted to perform accurate phonetic alignment.

p	o	r	-	r	e
p	o	n	e	r	-

Table 1: An example alignment of two cognates.

Alignment algorithms usually contain two main components: a metric for measuring distance between phones, and a procedure for finding the optimal alignment. The former is often calculated on the basis of phonological features that encode certain properties of phones. An obvious candidate for the latter is a well-known dynamic programming (DP) algorithm for string alignment (Wagner and Fischer, 1974), although other algorithms can used as well. The task of finding the optimal alignment is closely linked to the task of calculating the distance between two sequences. The basic DP algorithm accomplishes both tasks. Depending on the application, either of the results, or both, can be used.

In this paper, I present a new alignment algorithm for alignment of cognates. It combines various techniques developed for sequence comparison with an appropriate scoring scheme for computing phonetic similarity on the basis of multivalued features.

The new algorithm performs better, in terms of accuracy and efficiency, than comparable algorithms reported by Covington (1996) and Somers (1999). Although the main focus of this paper is diachronic phonology, the techniques proposed here can also be applied in other contexts where it is necessary to align phonetic sequences.

2 Background

2.1 Cognates, Segments, and Features

Phonology is the study of sound patterns in language. It is one of the major areas of linguistics. The area is usually divided into synchronic and diachronic phonology. The former deals with languages as they exist at one point in time, while the latter is concerned with language development through time.

Languages are genetically related if they derive from a common ancestor. For example, English, German and Swedish are related because they all derive from Proto-Germanic. Such languages always share a certain number of cognates, or words that derive from a single word in the ancestor language. For example, English *hundred*, French *cent*, and Polish *sto* are all cognates, descendants of the Proto-Indo-European **kmtom* (an asterisk denotes a reconstructed form).

Phonetic segments, or *phones*, are usually divided into two basic categories: *consonants*, which involve some type of obstruction in the vocal tract, and *vowels*, which are made with a very open vocal tract. Consonants are usually classified according to the place of articulation (labial, dental, palatal, etc.), and/or the manner of articulation (stops, fricatives, nasals, etc.). Table 2 contains a summary of English consonants. Vowels have three basic articulatory qualities: *height*, *backness*, and *rounding*. For example, the vowel that occurs in the English word *rat* is identified as low, front, and unrounded. *Glides* are consonants that are phonetically similar to vowels. English has two glides: /y/ as in *yell*, and /w/ as in *well*.

Features are units of speech, smaller than a segment, describing an aspect of articulation. Features can be *unary*, *binary*, or *multivalued*, depending on the number

	Stop	Fricative	Affricate	Nasal	Liquid	Glide
Bilabial	p b			m		
Labio-dental		f v				
Dental		θ ð				
Alveolar	t d	s z		n	l	
Palato-alveolar		ʃ ʒ	tʃ dʒ			
Retroflex					r	
Palatal						y
Velar	k g			ŋ		
Labio-velar						w

Table 2: Summary of English consonants.

of possible values. Every speech sound can be characterized by specifying a vector of feature values, but the characterization depends on the set of features that has been selected. A well known system of binary features is the one devised by Chomsky and Halle (1968). A binary feature has exactly two values, a 'plus' value and a 'minus' value. For example, a binary feature specification for bilabial voiced stop /b/ includes [+voice], [–coronal], and [–continuant]. Feature vectors of a number of sounds are given in Table 6 on page 9.

2.2 Sequence Comparison

Sequence comparison is a term used to describe a number of techniques and algorithms for comparing two or more sequences of elements drawn from some alphabet. The algorithms have been successfully applied to problems in such diverse fields as molecular biology, speech recognition and gas chromatography (Kruskal, 1983).

Typically, one of two cases applies: a) one of the sequences being analyzed is a product of applying a certain number of elementary operations to the other sequence, or b) both sequences represent variants of some underlying sequence. The operations which are considered are usually substitutions and insertions/deletions (*indels*).[2] Less often the differences are described in terms of compressions/expansions and transpositions.

[2]A substitution can also be analyzed as a deletion-insertion pair.

In general, the number of possible alignments grows exponentially with the length of sequences. Usually, we are interested in finding the alignment that minimizes a certain numerical function called *edit distance*. For example, we can define the distance as the minimum number of substitutions and indels necessary to convert one sequence into another (the so-called Levenshtein distance). A good distance function should satisfy the following *metric axioms*:

1. $\forall a, b : d(a, b) \geq 0$ (*nonnegative property*)
2. $d(a, b) = 0 \Leftrightarrow a = b$ (*zero property*)
3. $d(a, b) = d(b, a)$ (*symmetry*)
4. $d(a, b) + d(b, c) \geq d(a, c)$ (*triangle inequality*)

3 Previous Work

In this section, I briefly describe several different approaches to phonetic alignment that have been reported within the last few years.

Covington (1996) developed an algorithm for the alignment of cognates on the basis of phonetic similarity. In a follow-up paper (1998), he extended the algorithm to align words from more than two languages. His algorithm consists of a specially designed evaluation metric and a guided search procedure for finding the minimal-cost alignment. The cost of a particular alignment is calculated by adding the penalties for corresponding segments. The evaluation metric is a function that defines the penalty for every pair of segments (Table 4 on page 7). The metric also specifies the penalty for "skips", that is the pairs in which one of the segments is a null. The values of the metric range from 0 for two identical consonants to 100 for two segments with no discernible similarity. I discuss Covington's approach in more detail in section 4.1.

Somers (1998) proposed a special algorithm for aligning children's articulation data with the adult model. He implemented three versions of the algorithm, which use different methods to compute the cost of substitution: the 'CAT' version based on binary articulatory features, the 'FS/P' version based on perceptual features, and the 'Lad' version based on multivalued features. There is no explicit penalty for indels. The algorithm, which depends heavily on the alignment of stressed vowels, is described in (Somers, 1999). Somers observes that "alignment can become somewhat arbitrary" in the cases where the compared sequences are not very similar. After running 'CAT' on Covington's test data, he concludes that, in terms of accuracy, it is as good as Covington's algorithm.

Gildea and Jurafsky (1996) align phonetic strings in their transducer induction system. The system induces phonological rules directly from a large corpus of corresponding underlying and surface word-forms. The authors found that a pre-alignment of the corresponding forms greatly improves the performance of the system. The pre-alignment is performed using the DP algorithm and a metric that is based on 26 binary features. The cost of substitutions is a straightforward Hamming distance between two feature vectors. The cost of indels is set at one quarter of the maximum possible substitution cost. Because the surface forms are generated directly from the underlying forms by the application of a few simple phonological rules, the alignment algorithm need not be sophisticated.

Nerbonne and Heeringa (1996; 1997; 2000) investigate the problem of measuring phonetic distance between dialects. The distance between two dialects is estimated by taking the sum of Levenshtein distances between two sets of corresponding words. They experimented with at least two different feature systems, but they don't provide the details of the systems. The cost of indels is set at half the average of all substitutions. The individual distances are calculated using the DP algorithm. The computed distance is normalized by dividing its value by the length of the longer word. Realizing that "not all features are equally important in classifying the sounds", they tried weighting each feature by information gain, but found it had an adverse effect on the quality of the alignments. They also concluded that the Manhattan distance is preferable to both Euclidean distance and Pearson correlation.

Some characteristics of the approaches described in this section are juxtaposed in Table 3.

Algorithm	Calculation of alignment	Calculation of distance	Dynamic programming	Phonological features
Covington (1996)	explicit	implicit	no	no
Somers (1998)	explicit	no	no	multivalued
Nerbonne & Heeringa (1997)	implicit	explicit	yes	binary
Gildea & Jurafsky (1996)	explicit	implicit	yes	binary

Table 3: Comparison of alignment algorithms

4 Comparing Phones

To align phonetic sequences, we first need a function for calculating the distance between individual phones. The numerical value assigned by the function to a pair of segments is referred to as the cost, or penalty, of substitution. The function is often extended to cover pairs consisting of a segment and the null character, which correspond to the operations of insertion and deletion (also called *indels*). The distance function is of crucial importance because it determines the relative evaluation of alternative alignments.

4.1 Covington's Metric vs. Feature-Based Metrics

Covington's evaluation metric is reproduced in Table 4. The metric is very simple; it uses no phonological features and distinguishes only three types of segments: consonants, vowels, and glides. Many important characteristics of sounds, such as place or manner of articulation, are ignored. For example, both *yacht* and *will* are treated identically as a glide-vowel-consonant sequence. Such a "low resolution" is definitely a deficiency of the metric. Another reservation hat can be raised with respect to the metric is that there is no theoretical basis for the values assigned to various cases. The metric was developed by trial and error using a test set of 82 cognate pairs from various related languages. Covington (1998) acknowledges that his metric is "just a stand-in for a more sophisticated, perhaps feature-based, system".

Penalty	Conditions
0	Exact match of consonants or glides (w, y)
5	Exact match of vowels (reflecting the fact that the aligner should prefer to match consonants rather than vowels if it must choose between the two)
10	Match of two vowels that differ only in length, or i and y, or u and w
30	Match of two dissimilar vowels
60	Match of two dissimilar consonants
100	Match of two segments with no discernible similarity
40	Skip preceded by another skip in the same word (reflecting the fact that affixes tend to be contiguous)
50	Skip not preceded by another skip in the same word

Table 4: Evaluation metric used by Covington (1996).

Contrary to its name, Covington's evaluation function is not a metric. Normally, the name "metric" is reserved for functions that satisfy the axioms of symmetry, triangle inequality, and the zero property (see section 2.2 above). It is immediately evident that the zero property is not satisfied, because two identical vowels are assigned a non-zero penalty. The triangle inequality is also violated in some cases; for example: $p(e, i) = 30$ and $p(i, y) = 10$, but $p(e, y) = 100$, where $p(x, y)$ is the penalty for aligning [x] with [y].

Both Gildea and Jurafsky (1996) and Nerbonne and Heeringa (1997) represent phonetic segments by binary vectors in which every entry stands for a single articulatory feature. Such metrics not only have the ability to distinguish a large number of different phones, but also stand on a much firmer theoretical base. The underlying assumption is that the number of binary features by which two given sounds differ is a good indication of their proximity. Moreover, phonological rules can often be expressed as a modification of a limited number of feature values. Therefore, sounds that differ in a small number of features are more likely to be related. An evaluation function defined as the Hamming distance between two feature vectors does satisfy all metric axioms.

	Covington's metric's clause	Covington's penalty	Average Hamming distance	Interpolated average distance
1	*"identical consonants or glides"*	0	0.0	0.0
2	*"identical vowels"*	5	0.0	0.0
3	*"vowel length difference only"*	10	1.0	12.4
4	*"non-identical vowels"*	30	2.2	27.3
5	*"non-identical consonants"*	60	4.81	58.1
6	*"no similarity"*	100	8.29	100.0

Table 5: The clause-by-clause comparison of Covington's metric (column 3) and a feature-based metric (columns 4 and 5).

It is interesting to compare the values of Covington's metric with the average Hamming distances produced by a feature-based metric. Since neither Gildea and Jurafsky (1996) and Nerbonne and Heeringa (1997) present their feature vectors in sufficient detail to perform the calculations, I adopted a fairly standard set of 17 binary features from Hartman (1981). Twenty five letters of the Latin alphabet (all but *q*) were taken to represent a sample set of most frequent phonemes. The feature vectors corresponding to the set are shown in Table 6. In order to handle all the phones in Covington's data set, two features were added: [tense] and [spread glottis].

The average feature distances between pairs of segments corresponding to every clause in Covington's metric are given in Table 5, next to Covington's "penalties". By definition, the Hamming distance between identical segments is zero. The distance between the segments covered by clause #3 is also constant and equal to one (the feature in question being [long] or [syllabic]). The remaining average feature distances were calculated using the sample set of phonemes. In order to facilitate comparison, the rightmost column of Table 5 contains the average distances interpolated between the minimum and the maximum value of Covington's metric.

The correlation between Covington's penalties and the average distances is very high (0.998), which demonstrates that the penalties in Covington's metric can be derived directly from the properties of sounds expressed in terms of features. Thus,

feature name	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	v	w	x	y	z
[tense]	+	−	−	−	+	−	−	−	+	−	−	−	−	−	+	−	−	−	−	+	−	+	−	+	−
[spread glottis]	−	−	−	−	−	−	−	+	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
[voice]	+	+	−	+	+	−	+	−	+	+	−	+	+	+	+	−	+	−	−	+	+	+	−	+	+
[back]	+	−	−	−	−	−	+	+	−	−	+	−	−	−	+	−	−	−	−	+	−	+	+	−	−
[coronal]	−	−	+	+	−	−	−	−	−	+	−	+	−	+	−	−	+	+	+	−	−	−	−	−	+
[continuant]	+	−	−	−	+	+	−	+	+	−	−	−	−	−	+	−	+	+	−	+	+	+	+	+	+
[high]	−	−	+	−	−	−	+	−	+	+	+	−	−	−	−	−	−	−	−	+	−	+	+	+	−
[strident]	−	−	+	−	−	+	−	−	−	+	−	−	−	−	−	−	−	+	−	−	+	−	−	−	+
[round]	−	−	−	−	−	−	−	−	−	−	−	−	−	−	+	−	−	−	−	+	−	+	−	−	−
[syllabic]	+	−	−	−	+	−	−	−	+	−	−	−	−	−	+	−	−	−	−	+	−	−	−	−	−
[obstruent]	−	+	+	+	−	+	+	+	−	+	+	−	−	−	−	+	−	+	+	−	+	−	+	−	+
[nasal]	−	−	−	−	−	−	−	−	−	−	−	−	+	+	−	−	−	−	−	−	−	−	−	−	−
[consonantal]	−	+	+	+	−	+	+	+	−	+	+	+	+	+	−	+	+	+	+	−	+	−	+	−	+
[low]	+	−	−	−	−	−	−	+	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
[anterior]	−	+	+	+	−	+	−	−	−	+	−	+	+	+	−	+	+	+	+	−	+	−	−	−	+
[distributed]	+	+	+	+	+	−	+	+	+	−	+	−	+	−	+	+	−	+	+	+	−	+	+	+	+
[delayed release]	−	−	+	−	−	−	−	−	−	+	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−

Table 6: Feature vectors adopted from Hartman (1981).

feature-based phonology provides a theoretical basis for Covington's manually constructed metric.

4.2 Binary vs. Multivalued Features

Although binary features are elegant and widely used, they might not be optimal for phonetic alignment. Their primary motivation is to classify phonological oppositions rather than to reflect the phonetic characteristics of sounds. In a strictly binary system, sounds that are similar often differ in a disproportionately large number of features. It can be argued that allowing features to have several possible values results in a more natural and phonetically adequate system. For example, there are many possible places of articulation, which form a near-continuum ranging from [labial] to [glottal].

Ladefoged (1995) devised a phonetically-based multivalued feature system. This system has been adapted by Connolly (1997) and implemented by Somers (1998). It contains about 20 features with values between 0 and 1. Some of them can take as many as ten different values (e.g. [place]), while others are basically binary oppositions (e.g. [nasal]). Table 8 on page 18 contains examples of multivalued features.

The main problem with both Somers's and Connolly's approaches is that they do not differentiate the weights, or *saliences*, that express the relative importance of individual features. For example, they assign the same salience to the feature [place] as to the feature [aspiration], which results in a smaller distance between [p] and [k] than between [p] and [p^h]. I found that in order to avoid such incongruous outcomes, the salience values need to be carefully differentiated; specifically, the features [place] and [stop] should be assigned significantly higher saliences than other features. The actual values used in my algorithm are given in section 6.1. The question of how to derive salience values in a principled manner is still open.

4.3 Similarity vs. Distance

Although all four algorithms listed in Table 3 measure relatedness between phones using a *distance* function, such an approach does not seem to be the best for dealing with phonetic units. I have shown in section 4.1 that Covington's evaluation function does not satisfy two important properties of metrics. However, those violations are not merely accidental; rather, they reflect certain inherent characteristics of phones. Since vowels are in general more volatile than consonants, the preference for matching identical consonants over identical vowels is justified. This insight cannot be expressed by a distance metric, which, by definition, assigns a constant, zero distance to all identical pairs of segments. Nor is it certain that the triangle inequality should hold for phonetic segments. A phone that has two different places of articulation, such as labio-velar [w], can be close to two phones that are distant from each other, such as labial [v] and velar [g].

In my algorithm, below, I employ an alternative approach to comparing segments, which is based on the notion of *similarity*. A similarity scoring scheme assigns large positive scores to pairs of related segments; large negative scores to pairs of dissimilar segments; and small negative scores to indels. The optimal alignment is the one that maximizes the overall score. Under the similarity approach, the score obtained by two identical segments does not have to be constant. Another important advantage of the similarity approach is the possibility of performing *local alignment* of phonetic sequences, which is discussed in section 5.4.

Ideally, we would like to have a general scoring scheme resembling Dayhoff's matrix for amino acids (Dayhoff, Baker and Hunt, 1983). However, it is not clear what kind of statistics could be used to establish individual scores. Also, the number of possible sounds is greater than the number of amino acids. The International Phonetic Alphabet, which is a standard for representing phonetic data, contains over 80 symbols, most of which can be modified by various diacritics. Assembling a matrix of such size by deriving each individual entry is not practicable. In the absence of such scoring matrix, a better solution is to calculate similarity scores using a set of multivalued features with appropriate salience coefficients.

5 Finding the Optimal Alignment

Once an appropriate function for measuring similarity between pairs of segments has been designed, we need an algorithm for finding the optimal alignment of phonetic sequences. While the DP algorithm, which operates in quadratic time, seems to be optimal for the task, both Somers and Covington opt for exhaustive search strategies. In my opinion, this is unwarranted.

5.1 The DP Algorithm

The basic algorithm (Wagner and Fischer, 1974) is based on the dynamic programming principle (Figure 1). Assume that we have two sequences a and b of length n and m, respectively. The distance is calculated recursively for successively larger subsequences. Let $D[i, j]$ denote the minimal distance between the initial subsequences of a and b containing i and j elements, respectively. $\delta(x, y)$ denotes the substitution cost between elements x and y. Similarly, $\delta(-, y)$ and $\delta(x, -)$ denote the insertion and deletion cost, respectively. The actual distance between the complete sequences can be found in $D[n, m]$ after the completion of the program. The optimal alignment can then be retrieved from D by starting at entry (n, m) and tracing back through the entries until entry $(0, 0)$ is reached. The algorithm requires $O(nm)$ time to calculate the minimal edit distance plus $O(n + m)$ time to determine the corresponding alignment.

```
D[0,0] := 0
for i := 1 to n do
        D[i,0] := D[i-1,0] + δ(a_i, -)
for j := 1 to m do
        D[0,j] := D[0,j-1] + δ(-, b_i)
for i := 1 to n do
        for j := 1 to m do
                D[i,j] = min(  D[i-1,j-1] + δ(a_i, b_j),
                               D[i-1,j] + δ(a_i, -),
                               D[i,j-1] + δ(-, b_i) )
```

Figure 1: The basic algorithm for computing the distance between two strings.

Since its discovery in the early 1970s, the basic alignment algorithm described above has been extended and modified to deal with a number of related issues in various disciplines. Some of those extensions are general enough to be applied to the problem of finding the best alignment of phonetic strings.

5.2 Tree Search vs. Dynamic Programming

The alignment problem is characterized by a small number of elements and a limited number of interactions between them. Unsurprisingly, applying a depth-first search to this problem results in the same operations being performed repeatedly in various branches of the tree. Why do Covington and Somers opt for using such a costly procedure even though they are aware that a more suitable algorithm exists?

Somers's algorithm is unusual because the selected alignment is not necessarily the one that minimizes the sum of distances between individual segments. Instead, it recursively selects the most similar segments, or "anchor points", in the sequences being compared. Such an approach has a serious flaw. Suppose that the sequences to be aligned are *tewos* and *divut*. Even though the corresponding segments are slightly different, the alignment is straightforward. However, an algorithm that looks for the best matching segments first, will erroneously align the two *t*'s. Because of its recursive nature, the algorithm has no chance of recovering from such an error.

Covington, who uses a straightforward depth-first search to find the optimal alignment, provides the following arguments for eschewing the DP algorithm.

> First, the strings being aligned are relatively short, so the efficiency of dynamic programming on long strings is not needed. Second, dynamic programming normally gives only one alignment for each pair of strings, but comparative reconstruction may need the n best alternatives, or all that meet some criterion. Third, the tree search algorithm lends itself to modification for special handling of metathesis or assimilation. (Covington, 1996)

Covington's arguments are not entirely convincing. The efficiency of the algorithm might not be relevant in the simple case of comparing two words, but if the algorithm is to be of practical use, it will have to operate on large bilingual wordlists. Moreover, combining the alignment algorithm with some sort of strategy for identifying cognates on the basis of phonetic similarity is likely to require comparing thousands of words against one another. Having a polynomially bound algorithm in the core of such a system is crucial. In any case, since the DP algorithm involves neither significantly larger overhead nor greater programming effort, there is no reason to avoid using it even for relatively small data sets.

In response to the second argument it can be pointed out that a simple modification the basic DP algorithm will produce all alignments that are within ϵ of the optimal distance (Myers, 1995). By applying methods from the operations research literature (Fox, 1973), the algorithm can be adapted to deliver the n best solutions. Both of these modifications preserve the algorithm's polynomial complexity.

The third argument is somewhat hypothetical because Covington's algorithm handles neither metathesis nor assimilation. It remains to be seen if incorporating the detection of these phenomena results in more accurate alignments. The basic set of operations (substitutions and indels) can, nevertheless, be extended to include both transpositions of adjacent segments (metathesis) (Lowrance and Wagner, 1975) and compressions and expansions (Oommen, 1995).

In conclusion, until it is clearly shown that the advantages of using depth-first search to find the best alignment outweigh its inefficiency, it seems preferable to use the dynamic programming algorithm for this purpose.

5.3 Affine Gap Functions

The motivation for generalized gap scores arises from the fact that in diachronic phonology not only individual segments but also entire morphemes and syllables are sometimes deleted. In order to take this fact into account, the penalty for a gap can be calculated as a function of its length, rather than as a simple sum of individual deletions. One solution is to use an *affine* function of the form $gap(x) = r + sx$, where r is the penalty for the introduction of a gap, and s is the penalty for each symbol in the gap.

Gotoh (1982) describes a method for incorporating affine gap scores into the DP alignment algorithm. Two additional arrays (AG and BG) record the best alignment that ends with a gap in the corresponding string. $AG[i, j]$ is interpreted as the minimal distance between the initial subsequences of a and b when the subsequence of a ends with a gap, Similarly, BG corresponds to the case when the subsequence of b ends with a gap. The body of the main loop in the modified algorithm is shown in Figure 2.

Gotoh is not totally categorical about the initial settings of AG and BG. I found

$$AG[i,j] = min(AG[i-1,j] + s, D[i-1,j] + (r+s))$$
$$BG[i,j] = min(BG[i,j-1] + s, D[i,j-1] + (r+s))$$
$$D[i,j] = min(D[i-1,j-1] + \delta(a_i,b_j), AG[i,j], BG[i,j])$$

Figure 2: The modification of the basic dynamic programming algorithm (Table 1) required to accommodate affine gap scores.

that $AG_{0,j}$ and $BG_{i,0}$ must be set to $+\infty$ for all values of i and j. In this way, we can be sure that the distance between two unrelated strings of length 1 is in fact equal to $2(r+s)$, as required.

Incidentally, Covington's penalties for indels can be expressed by an affine gap function. Recall that in his metric the penalty for an indel is 40 if it is preceded[3] by another indel, and 50 otherwise. This is equivalent to defining an affine gap function with $r = 10$ and $s = 40$.

5.4 Local and Semiglobal Comparison

Local comparison (Smith and Waterman, 1981) is made possible by using both positive and negative similarity scores. In local, as opposed to global, comparison, only similar subsequences are matched, rather than entire sequences. This often has the beneficial effect of separating inflectional and derivational affixes from the roots. Such affixes tend to make finding the proper alignment more difficult. It would be unreasonable to expect affixes to be stripped before applying the algorithm to the data, because one of the very reasons to use an automatic aligner is to avoid analyzing every word individually.

The basic DP algorithm can easily be modified to perform local comparison (Figure 3). The interpretation of $D[i,j]$ is changed to the minimal distance between *suffixes* of the initial subsequences of a and b containing i and j elements, respectively. The first row and the first column of D are initialized with zeroes. $D[i,j]$ is set to the minimum of four rather than three values, the fourth being zero. This

[3]Apparently, Covington uses the verb *precede* to mean *immediately precede*.

has an effect that no entry of D is a negative number. A zero entry is interpreted as the alignment between the empty suffixes of the initial subsequences. The optimal distance is no longer found in $D[n, m]$, but rather it is equal to the maximum entry in D. The optimal alignment is retrieved by tracing back through the array, starting at the maximum entry and stopping as soon as a zero entry is found.

```
1     D[0,0] := 0
2     for i := 1 to n do
3         D[i,0] := 0
4     for j := 1 to m do
5         D[0,j] := 0
6     for i := 1 to n do
7         for j := 1 to m do
8             D[i,j] = min(  D[i-1,j-1] + δ(a_i, b_j),
9                            D[i-1,j] + δ(a_i, -),
10                           D[i,j-1] + δ(-, b_i) ,
11                           0 )
```

Figure 3: The basic algorithm for computing local similarities between two strings.

Semiglobal comparison is a combination of local and global comparison. The idea is to assign a score of zero to both the leading and the trailing indels with the goal of ignoring spurious affixes. The practical effect for cognate alignment is that a spurious affix can be separated from one of the words being compared.

The modifications required to make the DP algorithm perform semiglobal comparison are straightforward. As in the case of local comparison, the first row and column of D have to be initialized with zeroes. However, the zero option (line 11 in Figure 3) is not added. The optimal distance is the maximum entry in the union of the last row and the last column of D. Starting from this entry, we can retrieve the optimal alignment using the usual method, stopping when we reach one of the zeroes in the first row or column.

6 Implementation and Results

6.1 ALINE

ALINE is a C++ program that incorporates many of the ideas discussed in previous sections. The user can choose between local, semiglobal, and global alignment. The behavior of the program can be changed by setting a number of other parameters, which include the value of ϵ for producing near-optimal alignments, the maximum scores for consonants and vowels, the cost of indels, and the salience coefficients for individual features.

ALINE represents phonetic segments as vectors of feature values. The features that are currently used by ALINE are shown in Table 7. The values are encoded as floating-point numbers in the range $[0.0, 1.0]$. In principle, the number of values for any individual feature is infinite. Four features that have more than two possible values are shown in Table 8. The numbers are based on the measurements performed by Ladefoged (1995). The remaining features have exactly two possible values, 0.0 and 1.0. A special feature 'Double', which has the same values as 'Place', indicates the second place of articulation. Thanks to its continuous nature, the system of features and their values can easily be adjusted and augmented.

Syllabic	5	Place	40	Stop	50
Voice	10	Nasal	10	Retroflex	10
Lateral	10	Aspirated	5	Long	1
High	5	Back	5	Round	5

Table 7: Features used in ALINE and their salience settings.

For the calculation of relatedness, only relevant features are taken into account. If both segments are vowels, the following features are considered: Syllabic, High, Back, Round, Long, Nasal, Retroflex. Otherwise, the following features are considered: Syllabic, Stop, Place, Voice, Nasal, Retroflex, Lateral, Aspirated. When dealing with double-articulation consonantal segments, only the nearest places of articulation are used.

Feature name	Phonological term	Numerical value
Place	[bilabial]	1.0
	[labiodental]	0.95
	[dental]	0.9
	[alveolar]	0.85
	[retroflex]	0.8
	[palato-alveolar]	0.75
	[palatal]	0.7
	[velar]	0.6
	[uvular]	0.5
	[pharyngeal]	0.3
	[glottal]	0.1
Stop	[stop]	1.0
	[affricate]	0.9
	[fricative]	0.8
	[approximant]	0.6
	[high vowel]	0.4
	[mid vowel]	0.2
	[low vowel]	0.0
High	[high]	1.0
	[mid]	0.5
	[low]	0.0
Back	[front]	1.0
	[central]	0.5
	[back]	0.0

Table 8: Multivalued features and their value settings.

The similarity scores are established by first calculating the distance between two segments on the basis of a set of multivalued features with appropriate salience coefficients, and then subtracting the distance from the constant maximum score. The score is further lowered by a constant decrement for every vowel taking part in the comparison. The compression/expansion score is calculated by subtracting the sum of the distances $d(a_i, b_j)$, and $d(a_{i+1}, b_j)$ from the maximum score for compression/expansion, where segments a_i, a_{i+1} of the word a are surmised to correspond to segment b_j of the word b. The score is lowered if one or more segments are vowels.

The maximum score for substitution, the maximum score for compression/expansion, the vowel decrement, and the indel penalty are all provided to the program as parameters. Their current default values are 35, 45, 10, and 10, respectively. The salience values can also be provided as parameters. The values shown in Table 7 are defaults that have been found to yield the best results.

6.2 Data Input

Representing phonetically transcribed words using only the ASCII characters is not a trivial problem. The International Phonetic Alphabet, which is a standard for representing phonetic data, contains over 80 symbols, most of which can be modified by various diacritics. Using a system that assigns a numerical value to every combination is possible, but would result in a totally opaque data, difficult to enter and maintain.

A usual solution is to devise an *ad-hoc* scheme in which a mixture of alphanumeric characters and punctuation marks are used to represent phonetic symbols. For example, Eastlack (1977), in his simulation of systematic sound change in Ibero-Romance, represents /ɔ/ by the numeral 9, and /ɲ/ by *N* followed by a comma. Such schemes may be adequate to represent sounds from a particular language or language family, but usually do not scale up very well when more diverse data needs to be encoded. What is needed is a system powerful enough to express all IPA symbols, yet transparent enough to allow visual verification of the data.

The phonetic data for ALINE is encoded using a scheme in which every phonetic symbol is represented by a single lowercase letter followed by zero or more uppercase letters. The initial lowercase letter is the base letter most similar to the sound represented by the phonetic symbol. The remaining uppercase letters stand for the features in which the represented sound differs from the sound defined by the base letter. For example, the French front nasal vowel /œ̃/, which occurs in the word *un*, is represented by *oFN*, where *F* stands for *front*, and *N* stands for *nasal*. Other codes are *C* (*central*), *A* (*aspirated*), *P* (*palatal*), etc. In practice, the ratio of the number of characters used for the encoding to the number of encoded phonetic symbols is usually below 1.2.

6.3 Results

The best alignments are obtained when local comparison is used. For example, when aligning English *grass* with Latin *grāmen*, it is important to match only the first three segments in each word; the remaining segments are unrelated. ALINE obviously does not know the particular etymologies, but it can make a guess because [s] and [m] are not very similar phonetically. The following three alignments were obtained with global, semiglobal, and local comparison, respectively. It is the last one that best captures the relationship between the cognates.

g r - - æ s	‖ g r æ s ‖	‖ g r æ ‖ s
g r ā m e n	‖ g r ā m ‖ *en*	‖ g r ā ‖ men

The operations of compression and expansion prove to be very useful in the case of complex correspondences. For example, in the alignment of Latin *factum* with Spanish *hecho*, the affricate [ʧ] should be linked with both [k] and [t] rather than with just one of them, because it originates from the merger of the two consonants. Note that taking a sequence of substitution and deletion as compression is unsatisfactory because it cannot be distinguished from an actual sequence of substitution and deletion. ALINE posits this operation particularly frequently in cases of diphthongization of vowels (see alignments in Appendix A).

Covington's data set of 82 cognate pairs provides a convenient test for the algorithm. His English/Latin set is particularly interesting, because these two languages are not closely related. Some of the alignments produced by Covington's algorithm and ALINE are shown in Table 9. ALINE accurately discards inflectional affixes in *piscis* and *flāre*. In *fish/piscis*, Covington's aligner produces four alternative alignments, while ALINE selects the correct one. Both algorithms are technically wrong on *tooth/dentis*, but this is hardly an error considering that only the information contained in the phonetic string is available to the aligners. On Covington's Spanish/French data, ALINE does not make any mistakes. Unlike Covington's aligner, it properly aligns [l] in *arbol* with the second [r] in *arbre*. On his English/German data, it selects the correct alignment in those cases where Covington's aligner produces two alternatives. In the final, mixed set, ALINE makes a single mistake in *daughter/thugatēr*, in which it posits a dropped prefix rather than a syncopated syllable; in all other cases, it is right on target. A complete set of alignments of Covington's 82 cognate pairs generated by ALINE is included in Appendix A. Overall, ALINE clearly performs better than Covington's aligner.

	Covington's alignments	*ALINE's alignments*
three : trēs	θ r i y	‖ θ r iy ‖
	t r ē s	‖ t r ē ‖ s
blow : flāre	b l - - o w	‖ b l o ‖ w
	f l ā r e -	‖ f l ā ‖ re
full : plēnus	f - - - u l	‖ f u l ‖
	p l ē n u s	‖ p - l ‖ ēnus
fish : piscis	f - - - i š	‖ f i š ‖
	p i s k i s	‖ p i s ‖ kis
I : ego	- - a y	‖ ay ‖
	e g o -	‖ e ‖ go
tooth : dentis	- - - t u w θ	‖ t uw θ ‖
	d e n t i - s	den ‖ t i s ‖

Table 9: Examples of alignments of English and Latin cognates.

Somers (1999) tests one version of his algorithm, 'CAT', on the same set of cognates. CAT employs binary, rather than multivalued, features. Another important characteristic is that it pre-aligns the stressed segments in both sequences. Since CAT distinguishes between individual consonants, in some cases it produces more accurate alignments than Covington's aligner. However, because of its pre-alignment strategy, it is guaranteed to produce wrong alignments in all cases when the stress has moved in one of the cognates. For example, in the Spanish/French pair *cabeza/cap*, it aligns

[p] with [θ] rather than [b] and fails to align the two [a]'s. One can imagine how confused it would get when operating on closely related languages that have different stress rules.[4] In contrast, ALINE does not even consider stress, which, in the context of diachronic phonology, is too volatile to depend on. Except the single case of *daughter/thugatēr*, ALINE produces better alignments than Somers's algorithm.

7 Conclusion

The main challenge of computational diachronic phonology is building a system that given wordlists representing two or more daughter languages would be able to perform a fully automatic reconstruction of an unrecorded proto-language. Such system would have to be able first to identify cognates, then to align them in order to identify regular correspondences, and finally to reconstruct the proto-forms by assigning proto-phonemes to the correspondences. The alignment of corresponding segments in potential cognates is therefore an essential step of the comparative method.

The goal of my current research is to combine the new alignment algorithm with a cognate identification procedure, and perhaps with other steps of the comparative method. The alignment of cognates is possible only after the pairs of words that are suspected of being cognate have been identified. Identification of cognates is, however, an even more difficult task than the alignment itself. Moreover, it is hardly feasible without some kind of pre-alignment between candidate lexemes. A high alignment score of two semantically similar words should indicate that they are related. Semantic similarity can be estimated by referring to a semantic network such as WordNet. An integrated cognate identification algorithm would take as input unordered wordlists from two or more related languages, and produce a list of aligned cognate pairs as output. Such an algorithm would be a step towards developing a fully automated language reconstruction system.

[4] For example, stress regularly falls on the initial syllable in Czech and on the penultimate syllable in Polish, while in Russian it can fall anywhere in the word.

Appendix

A Alignments obtained with Covington's test set

Spanish-French cognate pairs (I)

yo : je	‖	y	o	‖			
	‖	ž	ə	‖			
tu : tu	‖	t	u	‖			
	‖	t	ü	‖			
nosostros : nous	‖	n	o	‖	sotros		
	‖	n	u	‖			
quién : qui	‖	k	ye	‖	n		
	‖	k	i	‖			
qué : quoi	‖	k	e	‖			
	‖	k	wa	‖			
todos : tous	‖	t	o	‖	dos		
	‖	t	u	‖			
una : une	‖	u	n	‖	a		
	‖	ü	n	‖			
dos : deux	‖	d	o	‖	s		
	‖	d	ö	‖			
tres : troix	‖	t	r	e	‖	s	
	‖	t	r	wa	‖		
hombre : homme	‖	o	m	‖	bre		
	‖	o	m	‖			

Spanish-French cognate pairs (II)

árbol : arbre	‖	a	r	b	o	l	‖	
	‖	a	r	b	-	r	‖	ə
pluma : plume	‖	p	l	u	m	‖	a	
	‖	p	l	ü	m	‖		
cabeza : cap	‖	k	a	b	‖	eθa		
	‖	k	a	p	‖			
boca : bouche	‖	b	o	k	‖	a		
	‖	b	u	š	‖			
pie : pied	‖	p	y	e	‖			
	‖	p	y	e	‖			
corazón : coeur	‖	k	o	r	‖	aθon		
	‖	k	ö	r	‖			
ver : voir	‖	b	e	r	‖			
	‖	v	wa	r	‖			
venir : venir	‖	b	e	n	i	r	‖	
	‖	v	ə	n	i	r	‖	
decir : dire	‖	d	e	θ	i	r	‖	
	‖	d	-	-	i	r	‖	
pobre : pauvre	‖	p	o	b	r	e	‖	
	‖	p	o	v	r	ə	‖	

English-German cognate pairs (I)

this : dieses		‖	ð	i	s	‖	
	dī	‖	z	e	s	‖	
that : das	‖	ð	æ	t	‖		
	‖	d	a	s	‖		
what : was	‖	w	a	t	‖		
	‖	v	a	s	‖		
not : nicht	‖	n	a	-	t	‖	
	‖	n	i	x	t	‖	
long : lang	‖	l	o	ŋ	‖		
	‖	l	a	ŋ	‖		
man : Mann	‖	m	æ	n	‖		
	‖	m	a	n	‖		
flesh : Fleisch	‖	f	l	e	š	‖	
	‖	f	l	ay	š	‖	
blood : Blut	‖	b	l	ə	d	‖	
	‖	b	l	ū	t	‖	
feather : Feder	‖	f	e	ð	ə	r	‖
	‖	f	ē	d	ə	r	‖
hair : Haar	‖	h	æ	r	‖		
	‖	h	ā	r	‖		

English-German cognate pairs (II)

ear : Ohr	‖	i	r	‖			
	‖	ō	r	‖			
eye : Auge	‖	a	y	‖			
	‖	a	w	‖	gə		
nose : Nase	‖	n	ow	z	‖		
	‖	n	ā	z	‖	ə	
mouth : Mund	‖	m	aw	-	θ	‖	
	‖	m	u	n	t	‖	
tongue : Zunge		‖	t	ə	ŋ	‖	
	t	‖	s	u	ŋ	‖	ə
	‖	t	-	ə	ŋ	‖	
	‖	t	s	u	ŋ	‖	ə
foot : Fuß	‖	f	u	t	‖		
	‖	f	ū	s	‖		
knee : Knie		‖	n	iy	‖		
	k	‖	n	ī	‖		
hand ; Hand	‖	h	æ	n	d	‖	
	‖	h	a	n	t	‖	
heart : Herz	‖	h	a	r	t	‖	
	‖	h	e	r	t	‖	s
liver : Leber	‖	l	i	v	ə	r	‖
	‖	l	ē	b	ə	r	‖

English-Latin cognate pairs (I)

<table>
<tr><td>and : ante</td><td>||</td><td>æ</td><td>n</td><td>d</td><td>||</td><td></td><td></td></tr>
<tr><td></td><td>||</td><td>a</td><td>n</td><td>t</td><td>||</td><td>e</td><td></td></tr>
<tr><td>at : ad</td><td>||</td><td>æ</td><td>t</td><td>||</td><td></td><td></td><td></td></tr>
<tr><td></td><td>||</td><td>a</td><td>d</td><td>||</td><td></td><td></td><td></td></tr>
<tr><td>blow : flāre</td><td>||</td><td>b</td><td>l</td><td>o</td><td>||</td><td>w</td><td></td></tr>
<tr><td></td><td>||</td><td>f</td><td>l</td><td>ā</td><td>||</td><td>re</td><td></td></tr>
<tr><td>ear : auris</td><td>||</td><td>i</td><td>r</td><td>||</td><td></td><td></td><td></td></tr>
<tr><td></td><td>||</td><td>aw</td><td>r</td><td>||</td><td>is</td><td></td><td></td></tr>
<tr><td>eat : edere</td><td>||</td><td>iy</td><td>t</td><td>||</td><td></td><td></td><td></td></tr>
<tr><td></td><td>||</td><td>e</td><td>d</td><td>||</td><td>ere</td><td></td><td></td></tr>
<tr><td>fish : piscis</td><td>||</td><td>f</td><td>i</td><td>š</td><td>||</td><td></td><td></td></tr>
<tr><td></td><td>||</td><td>p</td><td>i</td><td>s</td><td>||</td><td>kis</td><td></td></tr>
<tr><td>flow : fluere</td><td>||</td><td>f</td><td>l</td><td>ow</td><td>||</td><td></td><td></td></tr>
<tr><td></td><td>||</td><td>f</td><td>l</td><td>u</td><td>||</td><td>ere</td><td></td></tr>
<tr><td>star : stēlla</td><td>||</td><td>s</td><td>t</td><td>a</td><td>r</td><td>||</td><td></td></tr>
<tr><td></td><td>||</td><td>s</td><td>t</td><td>ē</td><td>l</td><td>||</td><td>la</td></tr>
<tr><td>full : plēnus</td><td>||</td><td>f</td><td>u</td><td>l</td><td>||</td><td></td><td></td></tr>
<tr><td></td><td>||</td><td>p</td><td>-</td><td>l</td><td>||</td><td>ēnus</td><td></td></tr>
<tr><td>grass : grāmen</td><td>||</td><td>g</td><td>r</td><td>æ</td><td>||</td><td>s</td><td></td></tr>
<tr><td></td><td>||</td><td>g</td><td>r</td><td>ā</td><td>||</td><td>men</td><td></td></tr>
<tr><td>heart : cordis</td><td>||</td><td>h</td><td>a</td><td>r</td><td>t</td><td>||</td><td></td></tr>
<tr><td></td><td>||</td><td>k</td><td>o</td><td>r</td><td>d</td><td>||</td><td>is</td></tr>
<tr><td>horn : cornū</td><td>||</td><td>h</td><td>o</td><td>r</td><td>n</td><td>||</td><td></td></tr>
<tr><td></td><td>||</td><td>k</td><td>o</td><td>r</td><td>n</td><td>||</td><td>ū</td></tr>
<tr><td>I : ego</td><td>||</td><td>ay</td><td>||</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>||</td><td>e</td><td>||</td><td>go</td><td></td><td></td><td></td></tr>
</table>

English-Latin cognate pairs (II)

knee : genū		‖	n	i	‖	y			
	ge	‖	n	ū	‖				
mother : māter	‖	m	ə	ð	ə	r	‖		
	‖	m	ā	t	e	r	‖		
mountain : mōns	‖	m	aw	n	t	‖	ə	n	
	‖	m	ō	n	s	‖			
name : nōmen	‖	n	ey	m	‖				
	‖	n	ō	m	‖	en			
new : novus	‖	n	yu	w	‖				
	‖	n	o	w	‖	us			
one : ūnus	‖	wə	n	‖					
	‖	ū	n	‖	us				
round : rotundus	‖	r	a	-	w	n	d	‖	
	‖	r	o	t	u	n	d	‖	us
sew : suere	‖	s	ow	‖					
	‖	s	u	‖	ere				
sit : sēdere	‖	s	i	t	‖				
	‖	s	ē	d	‖	ere			
three : trēs	‖	θ	r	iy	‖				
	‖	t	r	ē	‖	s			
tooth : dentis		‖	t	uw	θ	‖			
	den	‖	t	i	s	‖			
thin : tenuis	‖	θ	i	n	‖				
	‖	t	e	n	‖	uis			

Fox-Menomini cognate pairs

kiinwaawa : kenuaq	‖	k	ī	n	w	ā	‖	wa						
	‖	k	e	n	u	a	‖	q						
niina : nenah	‖	n	ī	n	a	‖								
	‖	n	e	n	a	‖	h							
naapeewa : naapɛɛw	‖	n	ā	p	ē	w	‖	a						
	‖	n	ā	p	ɛ̄	w	‖							
waapimini : waapemen	‖	w	ā	p	i	m	i	n	‖	i				
	‖	w	ā	p	e	m	e	n	‖					
nameesa : namɛɛqs	‖	n	a	m	ē	-	s	‖	a					
	‖	n	a	m	ɛ̄	q	s	‖						
okimaawa : okeemaaw	‖	o	k	i	m	ā	w	‖	a					
	‖	o	k	ē	m	ā	w	‖						
šiišiipa : seeqsep	‖	š	ī	-	š	ī	p	‖	a					
	‖	s	ē	q	s	e	p	‖						
	‖	š	ī	š	ī	p	‖	a						
	‖	s	ē	qs	e	p	‖							
ahkohkwa : ahkɛɛh	‖	a	h	k	o	h	‖	kwa						
	‖	a	h	k	ɛ̄	h	‖							
pemaatesiweni : pemaateswen	‖	p	e	m	ā	t	e	s	i	w	e	n	‖	i
	‖	p	e	m	ā	t	e	s	e	w	e	n	‖	
asenya : aqsɛn	‖	a	-	s	e	n	‖	ya						
	‖	a	q	s	ɛ	n	‖							

Cognate pairs from other languages

didōmi : dō	di	$\|$	d	ō	$\|$	mi			
		$\|$	d	ō	$\|$				
thugatēr : Tochter	$\|$	t^h	u	g	a	t	ē	r	$\|$
	$\|$	t	o	x	-	t	ə	r	$\|$
daughter : thugatēr		$\|$	d	o	t	ə	r	$\|$	
	t^hu	$\|$	g	a	t	ē	r	$\|$	
ager : ajras	$\|$	a	g	e	r	$\|$			
	$\|$	a	ǰ	-	r	$\|$	as		
bharāmi : pherō	$\|$	b^h	a	r	ā	$\|$	mi		
	$\|$	p^h	e	r	o	$\|$			
centum : hekaton		$\|$	k	e	n	t	u	m	$\|$
	he	$\|$	k	a	-	t	o	n	$\|$
centum : satəm	$\|$	k	e	n	t	u	m	$\|$	
	$\|$	s	a	-	t	ə	m	$\|$	

References

Chomsky, N. and Halle, M. (1968). *The Sound Pattern of English.* New York: Harper & Row.

Connolly, J. H. (1997). Quantifying target-realization differences. *Clinical Linguistics & Phonetics*, 11:267–298.

Covington, M. A. (1996). An algorithm to align words for historical comparison. *Computational Linguistics*, 22(4):481–496.

Covington, M. A. (1998). Alignment of multiple languages for historical comparison. In *Proceedings of COLING-ACL'98: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 275–280.

Dayhoff, M. O., Baker, W. C., and Hunt, L. (1983). Establishing homologies in protein sequences. *Methods in Enzymology*, 91:524–545.

Eastlack, C. L. (1977). Iberochange: a program to simulate systematic sound change in Ibero-Romance. *Computers and the Humanities*, 11:81–88.

Fox, B. L. (1973). Calculating the Kth shortest paths. *INFOR – Canadian Journal of Operational Research and Information Processing*, 11(1):66–70.

Gildea, D. and Jurafsky, D. (1996). Learning bias and phonological-rule induction. *Computational Linguistics*, 22(4):497–530.

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708.

Hartman, S. L. (1981). A universal alphabet for experiments in comparative phonology. *Computers and the Humanities*, 15:75–82.

Kruskal, J. B. (1983). An overview of sequence comparison. In Sankoff, D. and Kruskal, J. B., editors, *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*, pages 1–44. Reading, Mass.: Addison-Wesley.

Ladefoged, P. (1995). *A Course in Phonetics.* New York: Harcourt Brace Jovanovich.

Lowrance, R. and Wagner, R. A. (1975). An extension of the string-to-string correction problem. *Journal of the Association for Computing Machinery*, 22:177–183.

Myers, E. W. (1995). Seeing conserved signals. In Lander, E. S. and Waterman, M. S., editors, *Calculating the Secrets of Life*, pages 56–89. Washington, D.C.: National Academy Press.

Nerbonne, J. and Heeringa, W. (1997). Measuring dialect distance phonetically. In *Proceedings of the Third Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON-97)*.

Nerbonne, J., Heeringa, W., and Kleiweg, P. (2000). Edit distance and dialect proximity. In Sankoff, D. and Kruskal, J. B., editors, *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*, pages 1–14. To appear. Available at `http://odur.let.rug.nl/~nerbonne/paper.html`.

Nerbonne, J., Heeringa, W., van den Hout, E., van der Kooi, P., Otten, S., and van de Vis, W. (1996). Phonetic distance between Dutch dialects. In Durieux, G., Daelemans, W., and Gillis, S., editors, *CLIN VI: Proceedings of the Sixth CLIN Meeting*, pages 185–202, Antwerp, Centre for Dutch Language and Speech (UIA).

Oommen, B. J. (1995). String alignment with substitution, insertion, deletion, squashing, and expansion operations. *Information Sciences*, 83:89–107.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular sequences. *Journal of Molecular Biology*, 147:195–197.

Somers, H. L. (1998). Similarity metrics for aligning children's articulation data. In *Proceedings of COLING-ACL'98: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 1227–1231.

Somers, H. L. (1999). Aligning phonetic segments for children's articulation assessment. *Computational Linguistics*, 25(2):267–275.

Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173.