Phonetic alignment and similarity

Grzegorz Kondrak
Department of Computing Science
University of Alberta
Edmonton, AB T6G 2E8, Canada
Phone: +1 780 492 1779
Fax: +1 780 492 1071
E-mail: kondrak@cs.ualberta.ca

April 4, 2003

Abstract. The computation of the optimal phonetic alignment and the phonetic similarity between words is an important step in many applications in computational phonology, including dialectometry. After discussing several related algorithms, I present a novel approach to the problem that employs a scoring scheme for computing phonetic similarity between phonetic segments on the basis of multivalued articulatory phonetic features. The scheme incorporates the key concept of feature salience, which is necessary to properly balance the importance of various features. The new algorithm combines several techniques developed for sequence comparison: an extended set of edit operations, local and semiglobal modes of alignment, and the capability of retrieving a set of near-optimal alignments. On a set of 82 cognate pairs, it performs better than comparable algorithms reported in the literature.

Keywords: cognates, dialects, features, phonetic alignment, phonetic similarity

1. Introduction

The ability to quantify the phonetic similarity between words is important in many applications in both diachronic and synchronic phonology, including dialectometry. (In most context, the notions of *word similarity* and *word distance* are interchangeable.) A recent study (Heeringa et al., 2002) confirms that word-based methods for dialect comparison perform better than corpus-based methods that ignore word-boundaries. Such methods usually estimate word similarity as a (weighted) sum of the similarity between corresponding phonetic segments, and therefore depend crucially on their correct alignment. In contrast with word similarity, which is a rather subjective notion, we can usually establish the correct alignment with a high degree of confidence. An objective evaluation is therefore easier for an alignment algorithm than for a similarity algorithm.

Phonetic alignment is often an objective in itself. Usually, the strings to be aligned 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. Alignment of phonetic strings 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. This is not entirely surprising considering that both words and molecular sequences are made of a limited set of segments that undergo evolutionary changes and splits.

Both the word similarity and the word alignment algorithms usually contain two main components: a metric for measuring distance between phonetic segments and a procedure for finding the optimal alignment. The former is often calculated on the basis of phonological features that encode certain properties of phonetic segments. An obvious candidate for the latter is a well-known algorithm for string alignment (Wagner and Fischer, 1974), which is based on the dynamic programming[1] principle. The algorithm simultaneously calculates the similarity between two strings and their optimal alignment. Depending on the application, either of the results, or both, can be used.

In this paper, I present a new approach to the alignment of phonetic strings, and compare it to several other approaches that have been reported in the literature. The new approach combines various techniques developed for sequence comparison with a scoring scheme for computing phonetic similarity on the basis of multivalued articulatory features. An evaluation on a set of cognates demonstrates that it performs better than comparable algorithms. The method is applicable not only to the alignment of cognates but also to any other contexts in which it is necessary to align phonetic strings.

2. Related algorithms

In this section, I review several algorithms for calculating the phonetic alignment and/or similarity that have been reported in the literature. Some properties of the algorithms are summarized in Table I. The label *explicit* identifies the intended function of the algorithm, while the label *implicit* marks the functionality that is present but not overtly used.

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 depth-first search procedure for finding the minimal-cost alignment. The evaluation metric is a function that specifies the substitution cost for every pair of segments, and a context-dependent insertion/deletion (*indel*) cost. The total cost of a particular alignment is calculated by summing the costs of all substitutions and indels. I discuss Covington's approach in more detail in Sections 3.2 and 5.1.

[1] Dynamic programming is a technique of efficiently solving problems by combining previously computed solutions to smaller sub-problems.

Table I. Comparison of phonetic alignment/similarity algorithms.

Algorithm	Calculation of alignment	Calculation of distance	Dynamic progr.	Phonological features
Covington (1996)	explicit	implicit	no	no
Somers (1998)	explicit	no	no	multivalued
Gildea & Jurafsky (1996)	explicit	implicit	yes	binary
Kessler (1995)	implicit	explicit	yes	multivalued
Nerbonne & Heeringa (1997)	implicit	explicit	yes	binary
Oakes (2000)	explicit	explicit	yes	multivalued

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). After running 'CAT' on Covington's test data, he concludes that, in terms of accuracy, it is as good as Covington's algorithm. In Section 3.1, I point out a weakness in Somers'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 forms greatly improves the performance of the system. Because the surface forms are generated directly from the underlying forms by the application of a few simple phonological rules, the pre-alignment algorithm need not be sophisticated. The evaluation metric is based on 26 binary features. The cost of substitutions is a straightforward Hamming distance[2] between two feature vectors. The cost of indels is set at one quarter of the maximum possible substitution cost.

Kessler (1995) tested several different approaches for computing distance between Irish dialects. The dialects were represented by wordlists, each containing about 50 concepts. The most sophisticated method employs twelve multivalued phonetic features. The numeric feature values are assigned arbitrarily, and all features are given the same weight. The distance between phonetic segments is calculated as the difference averaged across all twelve features. The cost of indels is not specified in the paper. Kessler found that

[2] Hamming distance between two vectors is the number of elements that need to be changed to obtain one vector from the other.

the feature-based method performed worse than a simpler phoneme-based method, which employed a binary identity function between phonemes.

Nerbonne and Heeringa (1997) investigated the problem of measuring phonetic distance between Dutch dialects. The distance between two dialects is estimated by taking the sum of Levenshtein distances[3] between two sets of corresponding words. The cost of indels is set at half the average of all substitutions. The computed distance is normalized by dividing its value by the length of the longer word. The authors found that, for measuring distance between phonemes on the basis of features, the Manhattan distance is preferable to both Euclidean distance and Pearson correlation

Oakes's (2000) program JAKARTA contains a phonetically-based alignment algorithm, whose ultimate purpose is the discovery of regular sound changes. An impressive array of edit operations covers a number of sound-change categories. The cost of all the above operations is uniformly set at 1, while the cost of the standard substitution and insertion/deletion is set at 2. The phonetic characteristics of sound are stored by means of just three features: place, manner, and voicing, of which the first two have more than two values. However, the similarity between phonetic segments is estimated by checking the identity of the feature values only; there is no notion of the relative distance between various places or manners of articulation. Distinct phonetic segments can have identical feature assignments.

3. Finding the optimal phonetic alignment

Given two strings of length n and m, the basic dynamic programming algorithm takes $O(nm)$ time to calculate the minimal edit distance plus $O(n+m)$ time to determine the corresponding alignment. Each element of the table D of size $(n+1) \times (m+1)$ holds the minimal distance between a pair of the initial substrings. The final element $D[n,m]$ contains the minimal distance between the entire input strings. The idea is to calculate each element of D on the basis of a few neighbouring elements. The optimal alignment can then be retrieved form D by tracing back through the elements until the root element $D[0,0]$ is reached.

The dynamic programming algorithm is fast and seems to be optimal for the task of aligning phonetic strings. Nevertheless, both Somers and Covington opt for other search strategies. In this section, I argue that this is unwarranted.

[3] Levenshtein distance is the minimum number of substitutions and insertions/deletions necessary to convert one string into another.

3.1. Greedy search is not enough

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 strings being compared. Such an approach has a serious flaw. Suppose that the strings to be aligned are *tewos* and *divut* (Table II). Even though the corresponding segments are slightly different, the alignment is straightforward. However, a greedy 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. Regardless of the method of choosing the anchor points, an algorithm that never backtracks is not guaranteed to find the optimal alignment.

Table II. A correct and an incorrect alignment of a hypothetical cognate pair.

t	e	w	o	s
d	i	v	u	t

-	-	-	-	t	e	w	o	s
d	i	v	u	t	-	-	-	-

Somers (2000) argues that his alignment algorithm works very well on the children's articulation data, where the stressed vowel is a reliable anchor point. This strategy is rather risky in the context of the alignment of cognates, where stress is too volatile to depend on. Even dialects of the same language may have different stress rules. For example, stress regularly falls on the penultimate syllable in most varieties of Polish, but on the initial syllable in the Tatra mountains dialect. Somers (1999) nevertheless applies his algorithm to the alignment of cognates. In Section 7, I will examine the alignments reported in that paper.

3.2. Exhaustive search is too much

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. Covington provides the following arguments for adopting depth-first search rather than a more efficient dynamic programming approach.

> 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 mod-

ification[4] for special handling of metathesis or assimilation. (Covington, 1996)

I am not convinced by Covington's arguments. If the algorithm is to be of practical use, it should be able to operate on large bilingual wordlists. Most words may be quite short, but *some* words happen to be rather long. For example, the vocabulary lists of Algonquian languages contain many words that are longer than 20 phonemes. In such cases, the number of possible alignments exceeds 3^{20}, according to Covington. Even with search-tree pruning, such a combinatorial explosion of the number of nodes is likely to cause a painful slow-down. Moreover, combining the alignment algorithm with some sort of strategy for measuring phonetic similarity between a number of dialects is likely to require comparing thousands of words against each other. Having a polynomially bound algorithm in the core of such a system is crucial. In any case, since the dynamic programming algorithm involves neither significantly larger overhead nor greater programming effort, there is no reason to avoid using it even for relatively small data sets.

The dynamic programming algorithm is not only considerably faster than tree search but also sufficiently flexible to accommodate the proposed modifications without compromising its polynomial complexity. In the following section, I demonstrate that it is possible to retrieve from the edit distance table D the set of k best alignments, or the set of alignments that are within ε of the optimal solution, and that the basic set of editing operations (substitutions and indels) can be augmented to include both transpositions of adjacent segments (metathesis) and compressions/expansions.

4. Extensions to the basic dynamic programming algorithm

In this section, I describe a number of extensions to the basic dynamic programming algorithm, which have been proposed primarily to address issues in DNA alignment, and I show their applicability to phonetic alignment.

4.1. Retrieving a set of best alignments

At times, it may be desirable to find a number of alternative alignments that are close to the optimum rather than a single best alignment. Myers (1995) describes a modification of the basic dynamic programming algorithm that produces all alignments that correspond to distances below the threshold score of $d+\varepsilon$, where d is the optimal distance. The alignments are retrieved recursively from the edit distance table D, with the current partial alignment maintained on a stack.

[4] Covington does not elaborate on the nature of the modification.

In order to find the k-best alignments, the edit distance table D can be viewed as a graph with nodes corresponding to the elements in the table, and the arc lengths set according to the edit distance function. A recently proposed algorithm (Eppstein, 1998) discovers the k-shortest paths connecting a pair of nodes in a directed acyclic graph in time $O(e+k)$, where e is the number of edges in the graph.

4.2. String similarity

An alternative way of evaluating the affinity of two strings is to measure their similarity, rather than the distance between them. The similarity of two strings is defined as the sum of the individual similarity scores between aligned segments. A similarity scoring scheme normally 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. The basic dynamic programming algorithm can be adapted to compute the similarity by simply modifying it to select the minimum, rather than the maximum, partial score.

The similarity approach is closely related to the distance approach. In fact, it is often possible to transform one into the other. An important advantage of the similarity approach is the possibility of performing *local alignment* of strings, which is discussed next.

4.3. Local and semiglobal alignment

Informally, the optimal *local alignment* (Smith and Waterman, 1981) of two strings is the highest scoring alignment of their substrings. This notion is particularly useful in applications where only certain regions of two strings exhibit high similarity. For example, the local alignment of Cree *a̅pakosi̅s* and Fox *wa̅pikono̅ha* ‘mouse’ (Table III) matches the roots of the words and leaves out the unrelated affixes. (Double bars delimit the aligned substrings.) Such an affix-stripping behaviour is impossible to achieve with global alignment.

It should be clear why the switch from distance to similarity is not just a trivial change of terminology. If we tried to identify corresponding substrings by minimizing distance, we would almost always end up with empty or identical substrings. This is because the distance between any substrings that are less than perfect matches will be greater than zero. In contrast, a well-designed similarity scheme which rewards good matches and penalizes poor matches will allow regions of similarity to achieve meaningful lengths.

Semiglobal alignment is intermediate between local and global alignment. The idea is to assign a similarity score of zero to any indels at the beginning or the end of the alignment. Unlike in local alignment, the unmatched substrings that do not contribute to the total score cannot occur simultaneously

Table III. Various kinds of alignment.

global:	‖	-	ā	p	a	k	o	s	ī	s	-	-	-	-	‖
	‖	w	ā	p	i	k	o	-	-	-	n	ō	h	a	‖
local:		‖	ā	p	a	k	o	‖	sīs						
	w	‖	ā	p	i	k	o	‖	nōha						
semiglobal:		‖	ā	p	a	k	o	s	ī	s	‖				
	w	‖	ā	p	i	k	o	-	-	-	‖	nōha			
half-local:	‖	-	ā	p	a	k	o	‖	sīs						
	‖	w	ā	p	i	k	o	‖	nōha						

in both strings. The practical effect for cognate alignment is that a spurious affix can be separated from only one of the words being compared. Note that the unaligned segments do not affect the similarity score of the two strings, which would be the case if global alignment was used instead.

Another possible combination of local and global alignment, which I decided to call *half-local alignment*, is useful in aligning cognates. It is designed to reflect the greater relative stability of the initial segments of words in comparison with their endings.

4.4. Affine gap functions

A *gap* is a consecutive number of indels in one of the two aligned strings. In some applications, the occurrence of a gap of length k is more probable than the occurrence of k isolated indels. 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 indels. 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 dynamic programming alignment algorithm. Incidentally, Covington's penalties for indels can be expressed by an affine gap function with $r = 10$ and $s = 40$.

4.5. Additional edit operations

In addition to substitution and insertion/deletion, another useful edit operation is compression/expansion, which aligns two contiguous segments of one string with a single segments of the other string. In the context of the alignment of cognates, the compression/expansion operation facilitates the expression of complex phoneme correspondences. For example, in the alignment of stems of Italian *latte* and Spanish *leche*, the rightmost alignment in Table IV is the most accurate. Note that emulating compression as a sequence of substitution and deletion is unsatisfactory because it cannot be distinguished from an actual sequence of substitution and deletion.

Table IV. An example of the compression/expansion edit operation.

l	a	t	t		l	a	t	t		l	a	tt
l	e	č	-		l	e	-	č		l	e	č

Oommen (1995) formally defines the string alignment algorithm that incorporates the compression/expansion operation. The operation of transposition of adjacent segments can also be integrated into the dynamic programming algorithms, much along the same lines as in the case of compression/deletion. The details of the necessary modifications are given in (Lowrance and Wagner, 1975) and (Oommen and Loke, 1997).

5. Comparing phonetic segments

The distance/similarity function is of crucial importance in the phonetic alignment. The numerical value assigned by the function to a pair of segments is referred to as the substitution cost (in the context of distance), or as the substitution score (in the context of similarity). The function can be extended to cover other edit operations, such as insertions/deletions and compressions/expansions. The most elementary distance function assigns a zero cost to identical segments and a unary cost to non-identical segments. Such a function is simple to implement, but will perform poorly on phonetic alignment. This section is concerned with the problem of designing a better function, which would encode the knowledge about universal characteristics of sounds.

5.1. FEATURE-BASED METRICS

Covington (1996), for his cognate alignment algorithm, constructed a special distance function. It was developed by trial and error on a test set of 82 cognate pairs from various related languages. The distance function 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, which implies that [m] and [h] are assumed to be as similar as [t] and [t^h], and both *yacht* and *will* are treated identically as a glide-vowel-consonant string. The function's values for substitutions, which range from 0 for two identical consonants to 100 for two segments with no discernible similarity, are listed in the "penalty" column in Table V. The penalty for an indel is 40 if it is preceded by another indel, and 50 otherwise. Covington (1998) acknowledges that his distance function is "just a stand-in for a more sophisticated, perhaps feature-based, system".

Although Covington calls his distance function an "evaluation metric", it does not satisfy all metric axioms. The zero property is not satisfied because the function's value for two identical vowels is greater than zero. Also, the triangle inequality does not hold in all cases.

Both Gildea and Jurafsky (1996) and Nerbonne and Heeringa (1997) base their distance functions on binary features. Phonetic segments are represented by binary vectors in which every element stands for a single articulatory feature. Such a representation allows one to distinguish a large number of phonetic segments. The distance between two segments can be defined as the Hamming distance between two feature vectors, that is, the number of binary features by which the two sounds differ. A distance function defined in such a way satisfies all metric axioms.

It is interesting to compare the values of Covington's distance function with the average Hamming distances produced by a feature-based metric. For the calculations, I adapted a fairly standard set of binary features from Hartman (1981), with the addition of two features: [tense] and [spread glottis]. Twenty-five letters of the Latin alphabet (all but *q*) were taken to represent a sample set of most frequent phonemes.

Table V shows Covington's "penalties" juxtaposed with the average feature distances between pairs of segments computed for every clause in Covington's metric. 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 25 phonemes. In order to facilitate comparison, the rightmost column of Table V contains the average distances rescaled between the minimum and the maximum value of Covington's metric.

The correlation between Covington's penalties and the average Hamming distances is very high (0.998), which demonstrates that feature-based phonology provides a theoretical basis for Covington's manually constructed distance function.

5.2. Similarity and distance

Although all algorithms listed in Table I measure relatedness between phones by means of a *distance* function, such an approach does not seem to be the best for dealing with phonetic segments. The fact that Covington's distance function is not a metric is not an accidental oversight; rather, it reflects certain inherent characteristics of phones. Since vowels are in general more volatile than consonants, the preference for matching identical consonants over iden-

Table V. The clause-by-clause comparison of Covington’s distance function and a feature-based distance function.

	Clause in Covington’s distance function	Covington’s penalty	Average Hamming distance	Rescaled 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

tical vowels is justified. This insight cannot be expressed by a metric, which, by definition, assigns a 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 [b] and velar [g].

In my approach, I employ the similarity-based approach to comparing segments (cf. section 4.2). The similarity score for two phonetic segments indicates how similar they are. 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 strings, which is discussed in section 4.3. In local, as opposed to global, alignment, only similar substrings are matched, rather than entire strings. 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.

5.3. 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 within a language rather than to reflect universal characteristics of sounds. In a strictly binary system, sounds that are similar often differ in a disproportionately large number of features. For instance, [y], which is the initial sound of the word *you*, and [ʤ], which is the initial sound of the word *Jew*, have an astounding nine contrasting feature values; yet the sounds are close enough to be habitually confused by speakers whose first language is Spanish. 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 (1975) devised a phonetically-based multivalued feature system. This system was adapted by Connolly (1997) and implemented by Somers (1998). It contains about twenty articulatory features, some of which, such as *Place*, can take as many as ten different values, while others, such as *Nasal*, are basically binary oppositions. For example, the feature *Voice* has five possible values: [glottal stop], [laryngealized], [voice], [murmur], and [voiceless]. Feature values are mapped to numerical values in the $[0,1]$ range.

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]. In my opinion, in order to avoid such incongruous outcomes, the salience values need to be carefully differentiated; specifically, the features *Place* and *Manner* should be assigned significantly higher saliences than other features.

Although there is no doubt that not all features are equally important in classifying sounds, the question of how to how to assign salience weights to features in a principled manner is still open. Nerbonne and Heeringa (1997) experimented with weighting each feature by information gain but found that it actually had a detrimental effect on the quality of alignments. Kessler (1995) mentions the uniform weighting of features as one of possible reasons for the poor performance of his feature-based similarity measure. Covington (1996) envisages "using multivariate statistical techniques and a set of known 'good' alignments" for calculating the relative importance of each feature, but provides no specific details.

In my opinion, it seems feasible to derive the saliences automatically from a large corpus of aligned cognates by adapting methods developed for molecular biology (Durbin et al., 1998). Unfortunately, such a representative training set is not readily available because the task of establishing the correct alignment of cognates by hand is very time-consuming. Moreover, any selection of the training data would bias the similarity function towards particular languages.

An important advantage of the feature-based metrics is a small number of parameters. It would be ideal to have, as stated by Kessler (1995) in his computational analysis of Irish dialects, "data telling how likely it is for one phone to turn into the other in the course of normal language change." Such universal scoring schemes exist in molecular biology under the name of Dayhoff's matrices for amino acids (Dayhoff et al., 1983). However, the amount of data available in dialectology is many orders of magnitude smaller than what has already been collected in genetics. Moreover, 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 substitution matrix of such size by deriving each individual element is not practicable. In the absence of a universal scoring scheme for pairs of phonetic segments, the calculation of similarity scores on the basis of articulatory phonetic features with salience coefficients is a good working solution.

6. ALINE

ALINE is an implementation of the phonetic alignment approach advocated in this paper. The program incorporates many of the ideas discussed in previous sections. Similarity rather than distance is used to determine a set of best local alignments that fall within ε of the optimal alignment. The set of operations contains insertions/deletions, substitutions, and expansions/compressions. but not transpositions, which have been judged too sporadic to justify their inclusion in the algorithm. Multivalued features are employed to calculate similarity of phonetic segments. Affine gap functions seem to make little difference in phonetic alignment when local comparison is used, so the algorithm makes no distinction between clustered and isolated indels.

ALINE is written in C++ and runs under Unix.[5] It accepts a list of word pairs from the standard input, and produces a list of alignments and their similarity scores on the standard output. The behavior of the program is controlled by command-line parameters: ε sets the threshold of acceptable near-optimal alignments; C_{skip}, C_{sub}, and C_{exp} are the maximum scores for indels, substitutions, and expansions, respectively; and C_{vwl} determines the relative weight of consonants and vowels; The default values are $\varepsilon = 0$, $C_{skip} = -10$, $C_{sub} = 35$, $C_{exp} = 45$, and $C_{vwl} = 10$. Although local comparison is the default, the program can be re-compiled to perform global and semiglobal alignment.

ALINE employs the dynamic programming approach to compute the similarity table using the σ scoring functions defined in Table VI. The best alignments are than retrieved recursively from the similarity table. Phonetic segments are encoded as vectors of feature values. The function $diff(p,q,f)$ returns the difference between segments p and q for a given feature f. For a more detailed description of ALINE, see (Kondrak, 2002).

[5] ALINE is publicly available at `http://www.cs.ualberta.ca/~kondrak/`.

Table VI. Scoring functions.

$$\sigma_{skip}(p) = C_{skip}$$

$$\sigma_{sub}(p,q) = C_{sub} - \delta(p,q) - V(p) - V(q)$$

$$\sigma_{exp}(p,q_1q_2) = C_{exp} - \delta(p,q_1) - \delta(p,q_2) - V(p) - max(V(q_1),V(q_2))$$

where

$$V(p) = \begin{cases} 0 & \text{if } p \text{ is a consonant} \\ C_{vwl} & \text{otherwise} \end{cases}$$

$$\delta(p,q) = \sum_{f \in R} \text{diff}(p,q,f) \times \text{salience}(f)$$

where

$$R = \begin{cases} R_C & \text{if } p \text{ or } q \text{ is a consonant} \\ R_V & \text{otherwise} \end{cases}$$

Table VII enumerates the features that are currently used by ALINE and their salience settings. R_V and R_C are feature sets fully specified in Table VII: R_V contains features relevant for comparing two vowels, while R_C contains features for comparing other segments. A special feature *Double*, which has the same possible values as *Place*, indicates the second place of articulation. When dealing with double-articulation consonantal segments, only the nearest places of articulation are used.

Feature values are encoded as floating-point numbers in the range $[0,1]$. The numerical values of four principal features listed in Table VIII are taken from Ladefoged (1975), who established them on the basis of experimental measurements of distances between vocal organs during speech production. The remaining features have exactly two possible values, 0.0 and 1.0. The fact that the scheme is based on articulatory phonetics does not necessarily imply that it is optimal for phonetic alignment. Similar feature schemes of Connolly (1997) and Kessler (1995) also employ discrete ordinal values scaled between 0 and 1. The former author incorporates and expands on Ladefoged's proposal, while the latter simply selects the values arbitrarily.

The salience values in Table VII and the default values of the command-line parameters have been established by trial and error on a small set of alignments that included the alignments of Covington (1996). By no means should they be considered as definitive, but rather as a starting point for future refinements. It is worth noting that assigning equal weight to all features, although superficially more elegant, does not address the problem of unequal relevance of features.

Table VII. Features used in ALINE and their salience settings.

Feature	Salience	R_C	R_V	Feature	Salience	R_C	R_V
Syllabic	5	+	+	Place	40	+	-
Voice	10	+	-	Nasal	10	+	+
Lateral	10	+	-	Aspirated	5	+	-
High	5	-	+	Back	5	-	+
Manner	50	+	-	Retroflex	10	+	+
Long	1	-	+	Round	5	-	+

Table VIII. Multivalued features and their values.

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.
Manner	*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.

The feature system proposed here is highly dynamic in the sense that the similarity matrix can be modified by changing feature saliences or numerical values within features. Such modifications are important as it would be unrealistic to expect a single set of values to be optimal for all types of languages. The flexibility of the system makes it possible to adapt the similarity matrix to the data.

7. Evaluation

For the evaluation, I adopted the set of 82 cognate pairs compiled by Covington (1996), which contains mainly words from English, German, French, Spanish, and Latin. In spite of some defects, Covington's set became something of a benchmark when Somers (1999), in order to demonstrate that his and Covington's alignments are of comparable quality, applied his algorithm to the set. In order to perform a fair and consistent comparison, I refrained from making any corrections in the set of cognates. Note that a program that performs well on aligning cognates across distinct languages is also likely to perform well on a relatively easier task of aligning words across dialects.

The evaluation involves the alignment algorithms of Covington (1996), Somers (1999), and Oakes (2000), as well as ALINE and an emulation of an algorithm based on binary features. Oakes's program JAKARTA has been provided by the author. I re-implemented Covington's aligner from the description given in his article, and verified that my version produces the same alignments. Somers's alignments were reconstructed from the description of the differences between his and Covington's results, complemented by my understanding of the behaviour of his algorithm. The "binary" program uses the basic dynamic programming algorithm and a distance metric based on the set of binary features adapted from Hartman (1981).

7.1. Qualitative evaluation

Some of the alignments produced by Covington's algorithm give clues about the weaknesses of his approach. In Spanish *arbol* and French *arbre*, his aligner fails to match [r] with [l]. The reason is that it has only a binary notion of identity or non-identity of consonants, without any gradation of similarity. This lack of discernment also causes an occasional proliferation of alternative alignments.

The version that Somers applied to the cognate data set (CAT) employs binary, rather than multivalued, features. Since CAT distinguishes between individual consonants, it sometimes produces more accurate alignments than Covington's aligner. However, because of its unconditional alignment of the stressed vowels, CAT is guaranteed to fail in all cases when the stress has moved in one of the cognates.

In spite of its comprehensive set of edit operations, Oakes's JAKARTA makes many elementary mistakes: it frequently aligns consonants with vowels, postulates unusual sound changes with no foundation, and has a tendency to align the shorter words with the suffixes of the longer words.

The program based on binary features makes two types of mistakes. First, it fails to align phonetic segments, such as [v] and [w] in English *what* and German *was*, that are quite similar but differ with respect to many binary features (eight in this case). Second, because of its global alignment strategy, when aligning words of different length, it has a tendency for postulating gaps of indels inside the shorter word.

With the exception of a few mistakes, ALINE does a good job both on closely and remotely related language pairs. In many cases, ALINE correctly discards inflectional affixes, posits the operation of compression/expansion to account for the cases of diphthongization of vowels, and produces a single, correct alignment where Covington's aligner vacillates between alternatives.

7.2. Quantitative evaluation

In order to make the comparison of alignment algorithms more rigorous, I constructed the set of true alignments ("gold standard") for Covington's set of cognates to the best of my knowledge. For the comparison, I adopted

Table IX. Evaluation of alignment algorithms on Covington's data set.

Subset	Number of pairs	Score				
		Covington	Somers	Oakes	Binary	Kondrak
Spanish–French	20	19.0	17.0	15.0	18.8	20.0
English–German	20	18.0	18.0	16.0	18.0	18.5
English–Latin	25	18.1	19.5	9.0	13.0	24.0
Fox–Menomini	10	9.0	9.0	9.0	9.3	9.5
Other	7	4.7	3.0	4.0	5.0	6.0
Total	82	68.8	66.5	53.0	64.2	78.0

a straightforward scoring scheme. One point is awarded for every correct *unique* alignment. In the cases of $k > 1$ alternative alignments, the score is $\frac{1}{k}$ if one of them is correct, and 0 otherwise. In order to make the playing field even, complex correspondences, such as compression/expansion, were treated as optional. The results of the manual evaluation are given in Table IX.

ALINE is a clear winner in the comparison, achieving over 95% accuracy. Somers's results are almost as good as Covington's, which, as Somers (1999) points out, "is a good result for CAT [...] considering that Covington's algorithm is aimed at dealing with this sort of data." The program based on binary features generates mostly accurate alignments for closely related languages, but falters on the difficult English–Latin cognates. Oakes's JAKARTA scores well below the rest.

8. Computing phonetic similarity with ALINE

Besides finding the optimal alignment, ALINE also produces an overall similarity score, which is the sum of the individual scores between corresponding phonetic segments. One way of normalizing the overall score returned by ALINE so that it falls in the range $[0, 1]$ is to divide it by the length of the longer word multiplied by the maximum possible similarity score between segments. The normalized similarity score can be used as a general phonetic word similarity measure.

A possible application of ALINE is in the estimation of the relative "closeness" between languages or dialects, Table X shows the average normalized phonetic similarity between cognates belonging to four Algonquian languages. The data was automatically extracted from an electronic version of an etymological dictionary (Hewson, 1993). Interestingly, the average similarity values given in Table X imply a different relationship between the languages than the total number of shared cognates.

Table X. The number of shared cognates and the average phonetic cognate similarity for four Algonquian languages (nouns only).

Languages		Number of cognates	Average similarity
Fox	Menomini	121	.607
Fox	Cree	130	.616
Fox	Ojibwa	136	.626
Menomini	Cree	239	.620
Menomini	Ojibwa	259	.590
Cree	Ojibwa	408	.699

The results of the evaluation described in the previous section show that, overall, ALINE produces better alignments than other algorithms. However, the evaluation was performed on a relatively small set of cognates. In the absence of a more comprehensive test set, a better form of evaluation would be to apply ALINE to a task on which its performance could be easily appraised. An example of such a task is the identification of cognates from a dictionary-type data, where a normalized phonetic similarity between two words serves as an indicator of the likelihood of cognation. In (Kondrak, 2002), I show that ALINE performs well on the cognate identification task.

9. Conclusion

I presented a novel approach to the alignment of phonetic strings. The phonetic similarity between phonetic segments is computed on the basis of multivalued articulatory features, under the assumption that sounds produced in a similar way are likely to correspond to each other. The features are weighted according to their relative importance. The optimal alignment is calculated using the dynamic programming algorithm that incorporates several enhancements including an extended set of edit operations and the capability of retrieving a set of near-optimal alignments. ALINE, the program that implements the new approach, is publicly available.

Apart from finding the optimal alignment, ALINE calculates an overall phonetic similarity score, which, after normalization by word length, can serve as a phonetic similarity measure. Thus, the similarity of any two words, not necessarily cognates, can be quickly computed. ALINE can therefore be directly applied to dialect classification by computing similarity between wordlists representing distinct dialects.

Although originally developed for a specific task of cognate identification, ALINE is grounded in general principles of articulatory phonetics. The program has since proved its usefulness on such diverse applications as identifying easily confusable drug names and evaluating the performance of speech recognizers. Since the alignment of cognates representing related languages is not fundamentally different from the alignment of corresponding words representing distinct dialects, it is hoped that ALINE will turn out to be an effective tool for dialectologists as well.

Acknowledgements

Thanks to Graeme Hirst, Elan Dresher, Steven Bird, Radford Neal, Suzanne Stevenson, and Kevin Knight for their comments regarding this work. The author was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) during his Ph.D. research at the University of Toronto.

References

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 the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*. pp. 275–280.

Dayhoff, M. O., W. C. Baker, and L. T. Hunt: 1983, 'Establishing homologies in protein sequences'. *Methods in Enzymology* **91**, 524–545.

Durbin, R., S. R. Eddy, A. Krogh, and G. Mitchison: 1998, *Biological sequence analysis*. Cambridge University Press.

Eppstein, D.: 1998, 'Finding the *k* shortest paths'. *SIAM Journal on Computing* **28**(2), 652–673.

Gildea, D. and D. Jurafsky: 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.

Heeringa, W., J. Nerbonne, and P. Kleiweg: 2002, 'Validating Dialect Comparison Methods'. In: W. Gaul and G. Ritter (eds.): *Classification, Automation, and New Media. Proceedings of the 24th Annual Conference of the Gesellschaft für Klassifikation e. V.* pp. 445–452.

Hewson, J.: 1993, *A computer-generated dictionary of proto-Algonquian*. Hull, Quebec: Canadian Museum of Civilization.

Kessler, B.: 1995, 'Computational dialectology in Irish Gaelic'. In: *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*. pp. 60–67.

Kondrak, G.: 2002, 'Algorithms for Language Reconstruction'. Ph.D. thesis, University of Toronto. Available at http://www.cs.ualberta.ca/∼kondrak.

Ladefoged, P.: 1975, *A Course in Phonetics*. New York: Harcourt Brace Jovanovich.

Lowrance, R. and R. A. Wagner: 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: E. S. Lander and M. S. Waterman (eds.): *Calculating the Secrets of Life*. Washington, D.C.: National Academy Press, pp. 56–89.

Nerbonne, J. and W. Heeringa: 1997, 'Measuring Dialect Distance Phonetically'. In: *Proceedings of the 3rd Meeting of the ACL Special Interest Group in Computational Phonology*.

Oakes, M. P.: 2000, 'Computer Estimation of Vocabulary in Protolanguage from Word Lists in Four Daughter Languages'. *Journal of Quantitative Linguistics* **7**(3), 233–243.

Oommen, B. J.: 1995, 'String Alignment With Substitution, Insertion, Deletion, Squashing, and Expansion Operations'. *Information Sciences* **83**, 89–107.

Oommen, B. J. and R. K. S. Loke: 1997, 'Pattern recognition of strings with substitutions, insertions, deletions and generalized transpositions'. *Pattern Recognition* **30**(5), 789–800.

Smith, T. F. and M. S. Waterman: 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 the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*. pp. 1227–1231.

Somers, H. L.: 1999, 'Aligning Phonetic Segments for Children's Articulation Assessment'. *Computational Linguistics* **25**(2), 267–275.

Somers, H. L.: 2000, 'Personal communication'.

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