An Algorithm to Align Words for Historical Comparison

Michael A. Covington[*]
The University of Georgia

The first step in applying the comparative method to a pair of words suspected of being cognate is to align the segments of each word that appear to correspond. Finding the right alignment may require searching. For example, Latin dō *'I give' lines up with the middle* dō *in Greek* didōmi, *not the initial* di.

This paper presents an algorithm for finding probably correct alignments on the basis of phonetic similarity. The algorithm consists of an evaluation metric and a guided search procedure. The search algorithm can be extended to implement special handling of metathesis, assimilation, or other phenomena that require looking ahead in the string, and can return any number of alignments that meet some criterion of goodness, not just the one best. It can serve as a front end to computer implementations of the comparative method.

1. The problem

The first step in applying the comparative method to a pair of words suspected of being cognate is to align the segments of each word that appear to correspond. This alignment step is not necessarily trivial. For example, the correct alignment of Latin *dō* with Greek *didōmi* is

```
- - d ō - -
d i d ō m i
```

and not

```
d ō - - - -
d i d ō m i
```

```
d - - ō - -
d i d ō m i
```

```
- - - - d ō
d i d ō m i
```

or numerous other possibilities. The segments of two words may be misaligned because of affixes (living or fossilized), reduplication, and sound changes that alter the number of segments, such as elision or monophthongization.

Alignment is a neglected part of the computerization of the comparative method. The computer programs developed by Frantz (1970), Hewson (1974), and Wimbish (1989) require the alignments to be specified in their input. The Reconstruction Engine of Lowe and Mazaudon (1994) requires the linguist to specify hypothetical sound changes and

[*] Artificial Intelligence Center, The University of Georgia, Athens, Georgia 30602-7415; email `mcovingt@ai.uga.edu`.

canonical syllable structure. The cognateness tester of Guy (1994) ignores the order of segments, matching any segment in one word with any segment in the other.

This paper presents a guided search algorithm for finding the best alignment of one word with another, where both words are given in a broad phonetic transcription. The algorithm compares surface forms and does not look for sound laws or phonological rules; it is meant to correspond to the linguist's first look at unfamiliar data. A prototype implementation has been built in Prolog and tested on a corpus of 82 known cognate pairs from various languages. Somewhat surprisingly, it needs little or no knowledge of phonology beyond the distinction between vowels, consonants, and glides.

2. Alignments

If the two words to be aligned are identical, the task of aligning them is trivial. In all other cases, the problem is one of *inexact string matching,* i.e., finding the alignment that minimizes the difference between the two words. A dynamic programing algorithm for inexact string matching is well known (Sankoff & Kruskal 1983, Ukkonen 1985, Waterman 1995), but I do not use it, for several reasons. 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. More about this later; first I need to sketch what the aligner is supposed to accomplish.

An alignment can be viewed as a way of stepping through two words concurrently, consuming all the segments of each. At each step, the aligner can perform either a MATCH or SKIP. A match is what happens when the aligner consumes a segment from each of the two words in a single step, thereby aligning the two segments with each other (whether or not they are phonologically similar). A skip is what happens when it consumes a segment from one word while leaving the other word alone. Thus, the alignment

```
a b c -
- b d e
```

is produced by skipping *a*, then matching *b* with *b*, then matching *c* with *d*, then skipping *e*. Here as elsewhere, hyphens in either string correspond to skipped segments in the other.[1]

The aligner is not allowed to perform, in succession, a skip on one string and then a skip on the other, because the result would be equivalent to a match (of possibly dissimilar segments). That is, of the three alignments

```
a b - c     a - b c     a b c
a - d c     a d - c     a d c
```

only the third one is permitted; pursuing all three would waste time because they are equivalent as far as linguistic claims are concerned. (Determining whether *b* and *d* actually correspond is a question of historical reconstruction, not of alignment.) I call this restriction the NO-ALTERNATING-SKIPS RULE.

1 Traditionally, the problem is formulated in terms of operations to turn one string into the other. Skips in string 1 and string 2 are called *deletions* and *insertions* respectively, and matches of dissimilar segments are called *substitutions*. This terminology is inappropriate for historical linguistics, since the ultimate goal is to derive the two strings from a common ancestor.

To identify the best alignment, the algorithm must assign a PENALTY (cost) to every skip or match. The best alignment is the one with the lowest total penalty. As a first approximation, we can use the following penalties:

0.0 for an exact match;

0.5 for aligning a vowel with a different vowel, or a consonant with a different consonant;

1.0 for a complete mismatch;

0.5 for a skip (so that two alternating skips — the disallowed case — would have the same penalty as the mismatch to which they are equivalent).

Then the possible alignments of Spanish *el* and French *le* (phonetically [lə]) are:

```
e l
l ə        2 complete mismatches       = 2.0

- e l
l ə -      2 skips + 1 vowel pair      = 1.5

e l -
- l ə      2 skips + 1 exact match     = 1.0
```

The third of these has the lowest penalty (and is the etymologically correct alignment).

3. The search space

Figure 1 shows, in the form of a tree, all of the moves that the aligner might try while attempting to align two three-letter words (English [hæz] and German [hat]). We know that these words correspond segment-by-segment,[2] but the aligner doesn't. It has to work through numerous alternatives in order to conclude that

```
h æ z
h a t
```

is indeed the best alignment.

The alignment algorithm is simply a depth-first search of this tree, beginning at the top of Figure 1. That is, at each position in the pair of input strings, the aligner tries first a match, then a skip on the first word, then a skip on the second, and computes all the consequences of each. After completing each alignment it backs up to the most recent untried alternative and tries a different one. "Dead ends" in the tree are places where further computation is blocked by the no-alternating-skip rule.

As should be evident, the search tree can be quite large even if the words being aligned are fairly short. Table 1 gives the number of possible alignments for words of various lengths; when both words are of length n, there are about 3^{n-1} alignments, not counting "dead ends." Without the no-alternating-skip rule, the number would be about $5^n/2$. Exact formulas are given at the end of this paper.

2 Actually, as an anonymous reviewer points out, the exact correspondence is between German *hat* and earlier English *hath*. The current English *-s* ending may be analogical. This does not affect the validity of the example because /t/ and /s/ are certainly in corresponding positions, regardless of their phonological history.

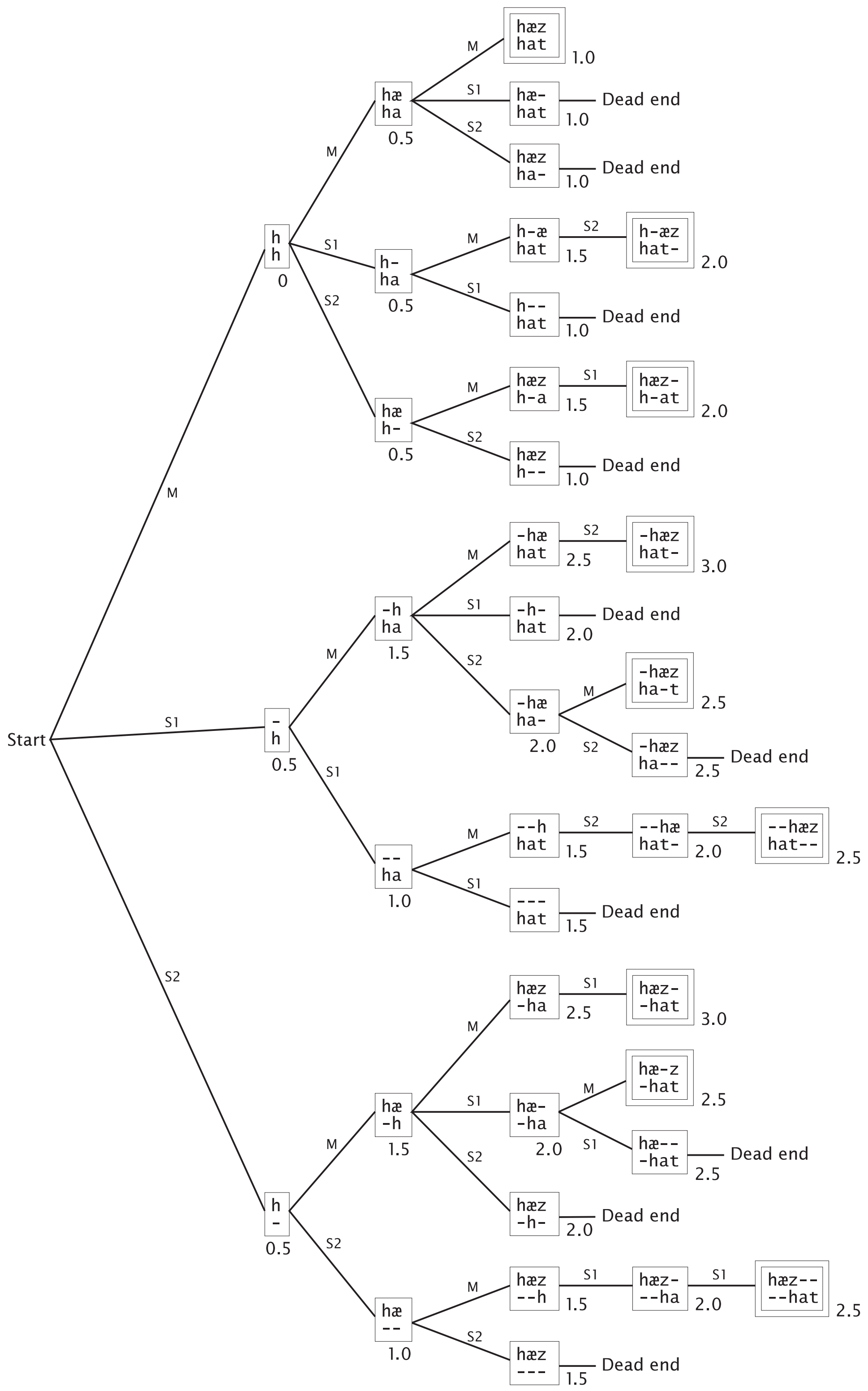

Figure 1
Search space for aligning English /hæz/ with German /hat/.

Fortunately, the aligner can greatly narrow the search by putting the evaluation metric to use as it works. The key idea is to abandon any branch of the search tree as soon as the accumulated penalty exceeds the total penalty of the best alignment found so far. Figure 2 shows the search tree after pruning according to this principle. The total amount of work is roughly cut in half. With larger trees, the saving can be even greater.

To ensure that a relatively good alignment is found early, it is important, at each stage, to try matches before trying skips. Otherwise the aligner would start by generating a large number of useless displacements of each string relative to the other, all of which have high penalties and do not narrow the search space much. Even so, the algorithm is quite able to skip affixes when appropriate. For example, when asked to align Greek *didōmi* with Latin *dō*, it tries only three alignments, of which the best two are:

```
d i d ō m i        d i d ō m i
d - - ō - -        - - d ō - -
```

Choosing the right one of these is then a task for the linguist rather than the alignment algorithm. However, it would be easy to modify the algorithm to use a lower penalty for skips at the beginning or end of a word than skips elsewhere; the algorithm would then be more willing to postulate prefixes and suffixes than infixes.

4. The full evaluation metric

Table 2 shows an evaluation metric developed by trial and error using the 82 cognate pairs shown in the subsequent tables. To avoid floating-point rounding errors, all penalties are integers, and the penalty for a complete mismatch is now 100 rather than 1.0. The principles that emerge are that syllabicity is paramount, consonants matter more than vowels, and affixes tend to be contiguous.

Somewhat surprisingly, it was not necessary to use information about place of articulation in this evaluation metric (although there are a few places where it might have helped). This accords with Anttila's (1989:230) observation that great phonetic subtlety is not needed to align words; what one wants to do is find the exact matches and align the syllabic peaks, matching segments of comparable syllabicity (vowels with vowels and consonants with consonants).

It follows that the input to the aligner should be in broad phonetic transcription, using symbols with closely similar values in both langauges. Excessively narrow phonetic transcriptions don't help; they introduce too many subtle mismatches that should have been ignored.

Phonemic transcriptions are acceptable insofar as they are also broad phonetic, but, unlike comparative reconstruction, alignment does not benefit by taking phonemes as the starting point. One reason is that alignment deals with syntagmatic rather than paradigmatic relations between sounds; what counts is the place of the sound in the word, not the place of the sound in the sound system. Another reason is that earlier and later languages are tied together more by the physical nature of the sounds than by the structure of the system. The physical sounds are handed down from earlier generations but the system of contrasts is constructed anew by every child learning to talk.

The aligner's only job is to line up words to maximize phonetic similarity. In the absence of known sound correspondences, it can do no more. Its purpose is to simulate a linguist's first look at unfamiliar data. Linguistic research is a bootstrapping process in which data leads to analysis and analysis leads to more and better-interpreted data. In its present form, the aligner does not participate in this process.

Table 1
Number of alignments as a function of lengths of words.

Lengths of words		Alignments
2	2	3
2	3	5
2	4	8
2	5	12
3	3	9
3	4	15
3	5	24
4	4	27
4	5	46
5	5	83
	⋮	
10	10	26,797

Table 2
Evaluation metric developed from actual data.

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

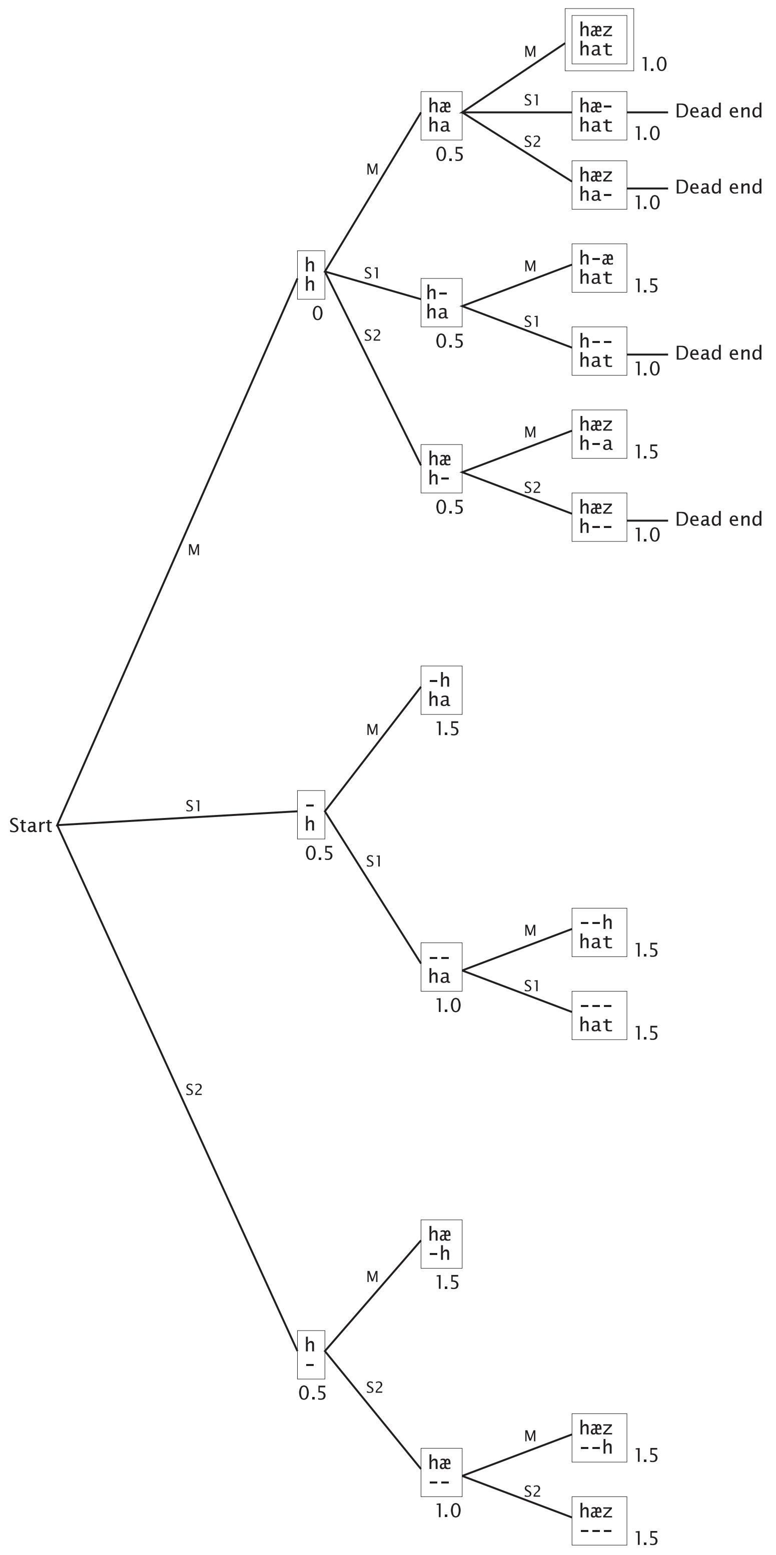

Figure 2
Same tree after pruning.

Table 3
Alignments obtained with test set of Spanish-French cognate pairs.

yo : *je* 'I'	y o ž ə
tu : *tu* 'you'	t u t ü
nosotros : *nous* 'you'	n o s o t r o s n u - - - - - -
quién : *qui* 'who?'	k y e n k i - -
qué : *quoi* 'what?'	k - e k w a
todos : *tous* 'all'	t o d o s t u - - -
una : *une* 'one' (f.sg.)	u n a ü n -
dos : *deux* 'two'	d o s d ö -
tres : *troix* 'three'	t r - e s t r w a -
hombre : *homme* 'man'	o m b r e o m - - -

5. Results on actual data

Tables 3–10 show how the aligner performed on 82 cognate pairs in various languages. (Tables 5–8 are loosely based on the Swadesh word lists of Ringe 1992.)[3]

These are "difficult" language pairs. On closely similar languages, such as Spanish / Italian or German / Danish, the aligner would have performed much better. Even so, on Spanish and French — chosen because they are historically close but phonologically very different — the aligner performed almost flawlessly (Tables 3–4). Its only clear mistake is that it missed the *l*:*r* correspondence in *arbre* : *árbol*, but so would the linguist without other data.

With English and German it did almost as well (Tables 5–6). The *s* in *this* is aligned with the wrong *s* in *dieses* because that alignment gave greater phonetic similarity; taking off the inflectional ending would have prevented this mistake. The alignments of *mouth* with *Mund* and *eye* with *Auge* gave the aligner some trouble; in each case it produced two alternatives, each getting part of the alignment right.

English and Latin (Tables 7–8) are much harder to pair up, since they are sepa-

3 To briefly address Ringe's main point: if the "best" alignment of a pair of words is used, the chance of finding a chance similarity is much higher than when using a fixed, canonical alignment.

Table 4
Alignments obtained with test set of Spanish-French cognate pairs (continued).

árbol : *arbre* ‘tree’	a r b - o l a r b r ə -
pluma : *plume* ‘feather’	p l u m a p l ü m -
cabeza ‘head’ : *cap* ‘promontory’	k a b e θ a k a p - - -
boca : *bouche* ‘mouth’	b o k a b u š -
pie : *pied* ‘foot’	p y e p y e
corazón : *coeur* ‘heart’	k o r a θ o n k ö r - - - -
ver : *voir* ‘see’	b - e r v w a r
venir : *venir* ‘come’	b e n i r v ə n i r
decir : *dire* ‘say’	d e θ i r d - - i r
pobre : *pauvre* ‘poor’	p o b r e p o v r ə

rated by millennia of phonological and morphological change, including Grimm’s Law. Nonetheless, the aligner did reasonably well with them, correctly aligning, for example, *star* with *stēlla* and *round* with *rotundus*. In some cases it was just plain wrong, e.g., aligning *tooth* with the *-tis* ending of *dentis*. In others it was indecisive; although it found the correct alignment of *fish* with *piscis*, it could not distinguish it from three alternatives. In all of these cases, eliminating the inflectional endings would have resulted in correct or nearly correct alignments.

Table 9 shows that the algorithm works well with non-Indo-European languages, in this case Fox and Menomini cognates chosen more or less randomly from Bloomfield (1941). Apart from some minor trouble with the suffix of the first item, the aligner had smooth sailing.

Finally, Table 10 shows how the aligner fared with some word pairs involving Latin, Greek, Sanskrit, and Avestan, again without knowledge of morphology. Because it knows nothing about place of articulation or Grimm’s Law, it can’t tell whether the *d* in *daughter* corresponds with the *th* or the *g* in Greek *thugatēr*. But on *centum* : *hekaton* and *centum* : *satəm* the aligner performed perfectly.

Table 5
Alignments obtained with test set of English-German cognate pairs.

this : *dieses*	ð i - - s d ī z ə s
that : *das*	ð æ t d a s
what : *was*	w̥ 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 a y 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

6. Improving the alignment algorithm

This alignment algorithm and its evaluation metric are, in effect, a formal reconstruction of something that historical linguists do intuitively. As such, they provide an empirical test of theories about how historical reconstruction is practiced.

There are limits to how well an aligner can perform, given that it knows nothing about comparative reconstruction or regularity of correspondences. Nonetheless, the present algorithm could be improved in several ways.

One obvious improvement would be to implement feature-based phonology. Implicitly, the aligner already uses two features, vocalicity and vowel length. A fuller set of features would have given a better alignment of *piscis* with *fish*, preferring *f*:*p* to *f*:*k*. Features are not all of equal importance for the evaluation metric; syllabicity, for instance, will surely be more important than nasality. Using multivariate statistical techniques and a set of known "good" alignments, the relative importance of each feature could be calculated.

Another improvement would be to enable the aligner to recognize assimilation, metathesis, and even reduplication, and assign lower penalties to them than to arbitrary mismatches. The need to do this is one reason for using tree search rather than the standard dynamic programming algorithm for inexact string matching. Dynamic

Table 6
Alignments obtained with test set of English-German cognate pairs (continued).

ear : *Ohr*	i r ō r	
eye : *Auge*	a - - y a w g ə	a y - - a w g ə
nose : *Nase*	n o w z - n ā - z ə	
mouth : *Mund*	m a w - θ m - u n t	m a w θ - m - u n t
tongue : *Zunge*	t - ə ŋ - t s u ŋ ə	
foot : *Fuß*	f u t f ū s	
knee : *Knie*	- n i y 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	

programming is, in effect, a breadth-first search of the tree in Figure 1; Ukkonen's (1985) improvement of it is a narrowed breadth-first search with iterative broadening. Both of these rely on computing parts of the tree first, then stringing partial solutions together to get a complete solution (that's what "dynamic programming" means). They do their partial computations in an order that precludes "looking ahead" along the string to undo an assimilation, metathesis, or reduplication. By contrast, my depth-first search algorithm can look ahead without difficulty.

Another crucial difference between my algorithm and dynamic programming is that, by altering the tree pruning criterion, my algorithm can easily generate, not just the best alignment or those that are tied for the best position, but the n best alignments, or all alignments that are sufficiently close to the best (by any computable criterion).

Multilateral alignments are needed when more than two languages are being compared at once. For example,

```
e l -
- l ə
i l -
```

is the etymologicaly correct three-way alignment of the masculine singular definite article in Spanish, French, and Italian. Multilateral alignments can be generated by aligning the second word with the first, then the third word with the second (and

Table 7
Alignments obtained with test set of English-Latin cognate pairs.

and : *ante*	æ n d			
	a n t e			
at : *ad*	æ t			
	a d			
blow : *flāre*	b l - - o w			
	f l ā r e -			
ear : *auris*	i - r - -			
	a w r i s			
eat : *edere*	i y t - - -			
	e - d e r e			
fish : *piscis*	- - - f i š	f - - - i š	f i - - - š	f i š - - -
	p i s k i s	p i s k i s	p i s k i s	p i s k i s
flow : *fluere*	f l o w - - -			
	f l - u e r e			
star : *stēlla*	s t a r - -			
	s t ē l l a			
full : *plēnus*	- - - f u l	f - - - u l		
	p l ē n u s	p l ē n u s		
grass : *grāmen*	g r - - æ s	g r æ - - s	g r æ s - -	
	g r ā m e n	g r ā m e n	g r ā m e n	
heart : *cordis* (gen.)	h a r - - t	h a r t - -		
	k o r d i s	k o r d i s		
horn : *cornū*	h o r n -			
	k o r n ū			
I : *ego*	- - a y			
	e g o -			

Table 8
Alignments obtained with test set of English-Latin cognate pairs (continued).

knee : *genū*	- - n i y	
	g e n ū -	
mother : *māter*	m ə ð ə r	
	m ā t e r	
mountain : *mōns*	m a w n t ə n	m a w n t ə n
	m ō - n - - s	m ō - n s - -
name : *nōmen*	n e y m - -	
	n ō - m e n	
new : *novus*	n y u w - -	n y u w -
	n - o w u s	n o w u s
one : *ūnus*	w ə n - -	
	- ū n u s	
round : *rotundus*	r a - w n d - -	
	r o t u n d u s	
sew : *suere*	s o w - - -	
	s - u e r e	
sit : *sēdere*	s i t - - -	
	s ē d e r e	
three : *trēs*	θ r i y	
	t r ē s	
tooth : *dentis* (gen.)	- - - t u w θ	
	d e n t i - s	
thin : *tenuis*	θ i n - - -	
	t e n u i s	

Table 9
Alignments obtained with test set of Fox-Menomini cognate pairs.

kiinwaawa : *kenuaq* ‘you (pl.)’	k ī n w ā w a - k e n - - u a q	k ī n w ā w a - k e n u - - a q
niina : *nenah* ‘I’	n ī n a - n e n a h	
naapeewa : *naapɛɛw* ‘man’	n ā p ē w a n ā p ɛ̄ w -	
waapimini : *waapemen* ‘maize’	w ā p i m i n i w ā p e m e n -	
nameesa : *namɛɛqs* ‘fish (n.)’	n a m ē - s a n a m ɛ̄ q s -	
okimaawa : *okeemaaw* ‘chief’	o k i m ā w a o k ē m ā w -	
šiišiipa : *seeqsep* ‘duck (n.)’	š ī - š ī p a s ē q s e p -	š ī š - ī p a s ē q s e p -
ahkohkwa : *ahkɛɛh* ‘kettle’	a h k o h k w a a h k ɛ̄ h - - -	
pemaatesiweni : *pemaatesewen* ‘life’	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* ‘stone (n.)’	a - s e n y a a q s ɛ n - -	

Table 10
Alignments obtained with cognate pairs from other languages.

Greek *didōmi* : Latin *dō* ‘I give’	d i d ō m i - - d ō - -	d i d ō m i d - - ō - -	
Greek *thugatēr* : German *Tochter* ‘daughter’	t^h u g a t ē r t o x - t ə r		
English *daughter* : Greek *thugatēr* ‘daughter’	- - d o t ə r t^h u g a t ē r	d - - o t ə r t^h u g a t ē r	d o - - t ə r t^h u g a t ē r
Latin *ager* : Sanskrit *ajras* ‘field’	a - g e r a ǰ r a s	a g - e r a ǰ r a s	a g e r - - a ǰ - r a s
Sanskrit *bharāmi* : Greek *pherō* ‘I carry’	b^h a r ā m i p^h e r - - ō	b^h a r ā m i p^h e r ō - -	
Latin *centum* : Greek *hekaton* ‘100’	- - k e n t u m h e k a - t o n		
Latin *centum* : Avestan *satəm* ‘100’	k e n t u m s a - t ə m		

implicitly also the first), and so on, but it would be advantageous to apply the evaluation metric to the whole set rather than just the pairs that are chained together. Multilateral alignment is also an important problem in DNA sequence analysis, and no general algorithm for it is known, but research is proceeding apace (Kececioglu 1993, Waterman 1995).

7. From here to the Comparative Method

Comparative reconstruction consists of three essential steps:

1. Align the segments in the (putative) cognates;
2. Find correspondence sets (corresponding to proto-allophones);
3. Identify some correspondence sets as phonetically conditioned variants of others (thereby reconstructing proto-phonemes).

Kay (1964) noted that the “right” set of alignments (of each of the cognate pairs) is the set that produces the smallest total number of sound correspondences. Steps 1 and 2 could therefore be automated by generating all possible alignments of all of the cognate pairs, then choosing the set of alignments that gives the fewest correspondence sets.

As Kay notes, this is not practical. Suppose the putative cognates are each 3 segments long. There are then 9 different alignments of each cognate pair, and if 100 cognate pairs are to be considered, there are $9^{100} \approx 2.65 \times 10^{95}$ sets of alignments to choose from, far too many to try on even the fastest computer.

However, a guided search along the same lines might well be worthwhile. First choose *one* alignment for each cognate pair — the best according to the evaluation metric, or if several are equally good, choose one arbitrarily. Construct the entire set of correspondence sets. Then go back and try one or two alternative alignments for each cognate pair, noting whether the size of the set of correspondence sets descreases. If so, adopt the new alignment instead of the previous one. For a set of 100 cognate pairs, this requires a total of only a few hundred steps, and the result should be close to the optimal solution. Reduction of correspondence sets to proto-phonemes is, of course, a separate task requiring a knowledge base of phonological features and information about phonetic plausibility.

Appendix: Size of the search space

The total number of alignments of a pair of words of lengths m and n can be calculated as follows.[4] Recall that a match consumes a segment of both words; a skip consumes a segment from one word but not the other. The complete alignment has to consume all the segments of both words. Accordingly, any alignment containing k matches must also contain $m-k$ skips on the first word and $n-k$ skips on the second word. The number of matches k in turn ranges from 0 to $\min(m,n)$. Thus, in general, the number of possible alignments is

$$\text{Alignments}(m,n) = \sum_{k=0}^{\min(m,n)} \text{number of alignments containing } k \text{ matches}$$

Without the no-alternate-skip rule, the number of alignments containing k matches is simply the number of ways of partitioning a set of $k+(m-k)+(n-k)=m+n-k$ moves into k matches, $m-k$ skips on word 1, and $n-k$ skips on word 2:

$$\text{Alignments}(m,n) = \sum_{k=0}^{\min(m,n)} \frac{(m+n-k)!}{k!(m-k)!(n-k)!}$$

(To give you an idea of the magnitude, this is close to $5^n/2$ for cases where $m=n$ and $n<20$ or so.)

With the no-alternate-skip rule, the number of alignments is exponentially smaller (about 3^{n-1} when $m=n$) and can be calculated from the recurrence relation

$$a(m,n) = a(m-1,n-1) + \sum_{i=0}^{n-2} a(m-1,i) + \sum_{i=0}^{m-2} a(i,n-1)$$

with the initial conditions $a(0,n)=a(m,0)=1$; for a derivation of this formula see Covington and Canfield (in preparation).

4 For assistance with mathematics here I am greatly indebted to E. Rodney Canfield. I also want to thank other mathematicians who offered helpful advice, among them John Kececioglu, Jeff Clark, Jan Willem Nienhuys, Oscar Lanzi III, Les Reid, and other participants in `sci.math` on the Internet.

References

Anttila, Raimo (1989) *Historical and comparative linguistics.* 2nd revised edition. (Amsterdam Studies in the Theory and History of Linguistic Science, IV: Current Issues in Linguistic Theory, 6.) Amsterdam: Benjamins.

Bloomfield, Leonard (1941) "Algonquian." *Linguistic Structures of Native America,* ed. C. Osgood, 85–129. (Viking Fund Publications in Anthropology, 6.) Reprint, New York: Johnson Reprint Corporation, 1963.

Covington, Michael A., and Canfield, E. Rodney (in preparation) *The number of distinct alignments of two strings.* Research report, Artificial Intelligence Center, The University of Georgia.

Frantz, Donald G. (1970) A PL/1 program to assist the comparative linguist. *Communications of the ACM* 13:353–356.

Guy, Jacques B. M. (1994) An algorithm for identifying cognates in bilingual wordlists and its applicability to machine translation. *Journal of Quantitative Linguistics* 1:35–42.

Hewson, John (1974) Comparative reconstruction on the computer. John M. Anderson and Charles Jones, eds., *Historical linguistics I: syntax, morphology, internal and comparative reconstruction,* 191–197. Amsterdam: North Holland.

Kay, Martin (1964) *The logic of cognate rcognition in historical linguistics.* (Memorandum RM-4224-PR.) Santa Monica: The RAND Corporation.

Kececioglu, John (1993) The maximum weight trace problem in multiple sequence alignment. *Combinatorial pattern matching: 4th annual symposium,* ed. A. Apostolico et al., 106–119. Berlin: Springer.

Lowe, John B., and Mazaudon, Martine (1994) The Reconstruction Engine: a computer implementation of the comparative method. *Computational Linguistics* 20:381–417.

Ringe, Donald A., Jr. (1992) *On calculating the factor of chance in language comparison.* Philadelphia: American Philosophical Society.

Sankoff, David, and Kruskal, Joseph B., eds. (1983) *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison.* Reading, Mass.: Addison-Wesley.

Ukkonen, Esko (1985) Algorithms for approximate string matching. *Information and Control* 64:100–118.

Waterman, Michael S. (1995) *Introduction to computational biology: maps, sequences and genomes.* London: Chapman & Hall.

Wimbish, John S. (1989) *WORDSURV: a program for analyzing language survey word lists.* Dallas: Summer Institute of Linguistics. Cited by Lowe and Mazaudon (1994).