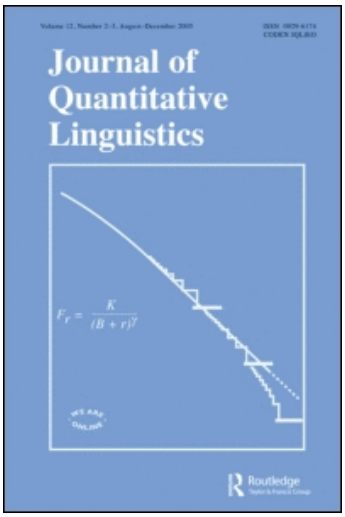

Journal of Quantitative Linguistics
Publication details, including instructions for authors and subscription information:
http://prod.informaworld.com/smpp/title~content=t716100702

Computer Estimation of Vocabulary in a Protolanguage from Word Lists in Four Daughter Languages

Michael P. Oakes

Online Publication Date: 01 December 2000

To cite this Article Oakes, Michael P.(2000)'Computer Estimation of Vocabulary in a Protolanguage from Word Lists in Four Daughter Languages',Journal of Quantitative Linguistics,7:3,233 — 243

To link to this Article: DOI: 10.1076/jqul.7.3.233.4105

URL: http://dx.doi.org/10.1076/jqul.7.3.233.4105

Journal of Quantitative Linguistics
2000, Vol. 7, No. 3, pp. 233–243

0929-6174/00/0703-0233$15.00

Computer Estimation of Vocabulary in a Protolanguage from Word Lists in Four Daughter Languages*

Michael P. Oakes
University of Sheffield, UK

ABSTRACT

Even if no written records of a protolanguage remain, it is possible to estimate what some of the words in that language might have been, by comparison of its reflexes in the more recent daughter languages. This method of protolanguage reconstruction is called the 'comparative method', and is described by Crowley (1992, Chapter 5). Although long practiced by human linguists, the comparative method is extremely time consuming, and only a few parts of the process have previously been automated. This paper describes a program which attempts to replicate the methodology described by Crowley almost in its entirety.

INTRODUCTION

Words in genetically related languages which appear to be derived from a common original form are said to be cognate with each other, and both are reflexes of the same form in the protolanguage or common ancestor language. Cognate words are vocabulary items which occur in two or more historically related languages, such that they have similar meanings, and one can be transformed into the other by a predictable series of phonological changes. Nothofer (1975) has manually produced tables showing the sound equivalences which occur in four languages spoken in or near the Indonesian island of Java, namely Javanese, Madurese, Malay and Sundanese, all of which originate from the ancestor language Proto-Malayo-Javanic (PMJ). One example of such an equivalence is Javanese d = Madurese jh = Malay j = Sundanese j = PMJ *z, as in the words for *road*, 'dalan', 'jhalan', 'jalan', 'jalan' and '*zalan' respectively. The asterisk before '*zalan' shows that this word is a hypothetical reconstruction, no longer heard today. The protophoneme *z may not have actually been pronounced 'z', but is the symbol used to denote the phoneme which gave rise to d, jh, j and j in the four modern languages. A system of sound correspondences was first described for Indo-European languages (Grimm's Law), and later shown to be found in all language families by Bloomfield (1925).

The task of identifying regular sound changes in bilingual wordlists has been described by Guy (1994) as follows:

> Given a sample word list from two related languages, extract the probable rules for predicting any word of one language from that of the other.

This task is a necessary precursor of protolanguage reconstruction. It also has applications in cognate identification for automatic text alignment in machine translation (Simard et al., 1992), and in the process of glottochronology, which is the use of statistical techniques to measure the degree of relatedness among cognate dialects (Lees, 1953). The word lists used in this study are the Javanese, Madurese, Malay and Sundanese words compiled by Nothofer (1975) for 195 meanings taken from Swadesh's (1964) list of 200 basic meanings.

A number of previous authors have tackled the problem of identifying regular sound changes in bilingual wordlists using the computer, but

*Address correspondence to: Michael P. Oakes, Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello Street, Sheffield S1 4DP, United Kingdom. Tel: 0114 222 1800. Fax: 0114 222 1810. E-mail: M.Oakes@dcs.shef.ac.uk

each of their methods solve only parts of the problem. Frantz's method (1970) requires the user to align the input word pairs, i.e., specify beforehand which characters correspond. Damerau (1975) considers only substitutions involving consonants. Damerau's method also depends on the assignment of a priori 'plausibility' weightings for phonemic changes which typically occur within the language family of interest, and thus (a) is not fully automatic and (b) is specific to a given language family rather than universally applicable. Lowe and Mazaudon's (1994) method does not discover new sound laws, but evaluates known or postulated ones. Guy's (1994) method works only for 1:1 character correspondences, where a single character in one language is replaced by a single character in the other. This means that the contexts in which character correspondences may occur are not discovered.

In summary, the parts of the task of automatic vocabulary reconstruction appear to be as follows:

(1) Record regular sound changes found in a given word list. A subtask of this is alignment at the character level. For example, if we are comparing the English 'four' with the German 'vier', we need to know that the 'f' corresponds to the 'v' and not to any other character. In this paper, a process called dynamic programming is used for character level alignment, but Guy initially considers all possible phoneme matches within a word pair, later using a variant of the chi square test to work out which sound matches are significant.
(2) Collate and statistically evaluate the discovered sound changes.
(3) Use the discovered rules to verify word pairs in real lists – or assume that the given cognate pairs are true and use these to evaluate the rules.
(4) Previous authors, given a modern language form and the transformation rules, have been able to reconstruct the vocabulary of sister or ancestor languages. In this paper we go one step further – by automating the comparative method, from two or more modern word lists, it is possible to calculate rules for transformation of the modern forms into the ancestor language.

In this paper we describe one computer program called JAKARTA, which performs steps (1) to (3) above, and a second program called PRAGUE which performs step (4).

DYNAMIC PROGRAMMING

Dynamic programming is a mathematical technique originally developed for solving optimisation problems in operational research (Norman, 1975). The technique has for some time been used for automatic spelling correction (Wagner & Fischer, 1974). The difference between two word forms (called the cost, or edit distance) is taken to be the smallest number of operations required to transform one word into the other. The allowable operations are substitution (a single character in one word is replaced by a single character in the other), deletion of a single character from the first word, and insertion of a single character into the second word. For example, in order to transform the Malay 'telur' into the Tagalog 'itlog' (both meaning *egg*), it is necessary to align the two words as follows: NULL = i (insertion), t = t, e = NULL (deletion), l = l, u = o (substitution), r = g (substitution). Thus four operations are required to transform 'telur' into 'itlog'. Other alignments are disregarded by dynamic programming because they involve a greater number of operations.

USE OF DYNAMIC PROGRAMMING TO ESTIMATE WHICH WORD PAIRS ARE COGNATE

The first task for the computer in identifying regular sound changes is to select which words are cognate for a given language pair. The method used is that of McEnery & Oakes (1996), where empirical data is used to determine the number of operations (insertions, deletions or substitutions) required to transform one of a pair of words into the other, above which the word pair is probably not cognate.

Word lists of the Swadesh meanings for each of the four daughter languages of PMJ were obtained, being those given by Nothofer (1975, p. 226), the first ten words of which are shown in Table 1. In one experiment, every word in the Malay list of 195 words was compared with every word in the Javanese list, whether or not they had the same meaning. A list was maintained of all word pairs where the number of operations required to transform one into the other was 0 (i.e., the two words were identical in appearance). In 81.5% of cases both members of the word pair had the same meaning in Swadesh's list. The remaining pairs where the two words had different meanings were mainly due to semantic drift, an example of which is the word 'kulit', which can mean either *skin* or *bark* in both Malay and Javanese. Similarly, a list was kept of all word pairs which differed by a single operation, and whether both words had the same meaning. Lists were also kept of those word pairs differing by each number of operations from two to six. The experiment was repeated for the language pairs Malay-Madurese and Malay-Sundanese, and the results are shown in Table 2. It was found that if a word pair differed by more than two operations, the chance of that word pair being cognate was virtually random. This suggests that a simple approximate method of selecting cognate pairs is to accept only those word pairs which differ by two or fewer operations, and this is the technique employed by program JAKARTA.

EXTENSIONS TO DYNAMIC PROGRAMMING

A number of authors have suggested extensions to the basic dynamic programming procedure

Table 1. Word Lists for the First Ten Swadesh Meanings.

English	Javanese	Madurese	Malay	Sundanese
All	Kabeh	Kabeh	Semua	Kabbhi
And	JiN	Lan	Dan	Ban
Animal	BinataN	Kewan	BinataN	Hewan
Ashes	Lebuq	Awu	Abu	LoloN
At	Diq	NoN	Di	E
Back	TukaN	Geger	BelakaN	Abali
Bad	GoreN	Eleq	Jahat	Jhubaq
Bark	Kulit	Kulet	Kulet	Koleq
Because	Kusabab	Sebab	Sebab	Sabab
Belly	BitiN	WetaN	Perut	Tabuq

N.B. The symbol N is used to represent the orthographic 'ng'.

Table 2. Percentage of True Cognates Obtained when the Malay Word List was Compared with Words in Three Other Languages for Different Dynamic Programming Edit Distances.

Operations	Javanese	Madurese	Sundanese
0	81.5	84.6	83.3
1	28.1	60.0	56.2
2	2.8	5.8	5.1
3	0.8	0.6	1.1
4	0.4	0.4	0.5
5	0.3	0.3	0.3
6	0.3	0.3	0.3

which were employed in the programs described in this paper.

In their work on bilingual sentence alignment, Gale and Church (1993) introduced additional operations into the dynamic programming algorithm. While the original algorithm allowed only for single character insertions, deletions and substitutions, Gale and Church also considered for example 2:1 correspondence, denoting that two sentences of one language correspond with just one sentence of the other. The six operations they used were 1:0, 0:1, 1:1, 2:1, 1:2 and 2:2. They also allowed for the fact that some operations are more commonly encountered in real data than others, by assigning higher costs to less frequently encountered operations. These ideas have be incorporated into both programs JAKARTA and PRAGUE.

Covington (1998) reports an example given originally by Mary Haas, using the languages Choctaw, Koasati and Cree, which shows that true alignments are best observed when families of languages rather than merely language pairs are considered, since the best trilingual alignment does not necessarily contain any of the best bilingual alignments.

Covington used a tree search rather than dynamic programming to find the best trilingual alignment, but a description of Dynamic Programming for words in three languages at once is given by Kruskal (1983, p. 33). Extending these ideas, program PRAGUE aligns words in all four daughter languages simultaneously.

DYNAMIC PROGRAMMING FOR IDENTIFYING REGULAR SOUND CHANGES IN BILINGUAL WORD LISTS

The first program described here, JAKARTA, uses the dynamic programming technique to compare the pairs of words found in a bilingual wordlist and keep a tally of the types of sound changes required to transform the vocabulary of one language into that of the other. Apart from simple insertion, deletion and substitution, the allowed operations correspond to the list of the types of sound change which typically occur between related languages throughout the world given by Crowley (1992, Chapter 2). In this way the identification of regular sound changes in the bilingual word list is linguistically motivated, but not restricted to a single language family. The types of sound change given by Crowley are as follows:

Lenition (1:1 correspondence): the progressive weakening of a sound, as in the sequence $-b \rightarrow p \rightarrow f \rightarrow h$;
Fortition (1:1): the reverse of lenition, the progressive strengthening of a sound;
Aphaeresis (1:0): the initial sound is dropped;
Apocope (1:0): the final sound is dropped;
Syncope (1:0): a sound in a medial position is dropped;
Cluster Reduction (3:2): a sound is deleted from a consonant cluster;
Excrescence (2:3): a consonant is added to a consonant pair to form a consonant cluster;
Epenthesis (2:3): a vowel is added in the middle of a word to break up two consonants in a cluster;
Prothesis (0:1): an initial sound is added;
Fusion (2:1): two originally separate sounds become a single sound, which shows some of the phonetic features of both of the original sounds;
Vowel Breaking (1:2); the reverse of fusion;
Assimilation (2:2, 2:1): two consonants become more similar to each other, e.g., $np \rightarrow mp$;
Dissimilation (2:2, 1:2): the reverse of assimilation.

Metathesis and hapology, although listed by Crowley, are not considered in this paper, both because they are relatively rare and are difficult to simulate using dynamic programming. Metathesis is the transposition of characters in a word, such as the t and s in 'tangis' and 'sangit' (Tagalog and Ilocano for *cry*). Hapology is the loss of a syllable followed or preceded by a similar sounding syllable.

Some of these operations can only take place if certain conditions hold true. The types of conditions imposed are:

(a) positional: e.g., prothesis must occur at the start of a word;
(b) certain characters must be vowels or consonants, e.g., vowel breaking can only involve vowels;

(c) certain corresponding characters involved in an operation must be identical, e.g., in transforming 'keldai' into 'keledai' (Malay and Indonesian for *donkey*), the epenthesis 'ld' → 'led' is allowed, since the initial and final characters are identical;
(d) adjacent characters must have at least one phonetic feature (such as place of articulation) in common, e.g., in assimilation;
(e) In the cases of lenition and fortition, corresponding characters must be the 'stronger' and 'weaker' equivalents of each other, as tabulated by Crowley (1992, p. 37).

A description of how the conditions for each type of sound change are programmed and the data which must be stored to ensure that the conditions hold true is given in the next section.

In order that any word string can still be transformed into any other word string, the standard dynamic programming operations of insertion, deletion and substitution are still allowed. For example, any 1:1 correspondence which cannot be accounted for by lenition or fortition is regarded as a substitution. Since the operations listed by Crowley are more likely to be found in real language pairs than the standard operations, the operations listed by Crowley are given a cost of just 1, while the standard operations are assigned a cost of 2. As with the basic dynamic programming algorithm, if a character is found to be identical in both words, the cost of aligning that character with itself is 0.

As program JAKARTA is run, the pairs of words in the bilingual word list are aligned in turn, to see which operations are required to transform each member of a cognate pair into the other. A tally is kept of each type of operation encountered, and the characters involved. If the cost of transforming one word of a pair into the other is more than a set threshold (4), the word pair is considered not cognate and disregarded. After all word pairs in the data set have been considered, all sound changes which have been encountered a set number of times are deemed to be regular sound changes for that language pair.

DATA STRUCTURES FOR THE RECOGNITION OF REGULAR SOUND CHANGES

The data which is stored by program JAKARTA in order to recognise sound changes of the types described by Crowley are as follows:

(1) A record is kept of which characters are consonants and which are vowels.
(2) A record is kept of the phonetic features possessed by each character, using the data on p. 15 of Crowley's book, where the phonetic alphabet is characterised by vocalisation or otherwise, place of articulation and manner of articulation, as follows:
   (a) Vocalised (yes or no);
   (b) Place (bilabial, labiodental, dental_alveolar, retroflex, alveolo_ palatal, velar, uvular, pharyngeal, glottal, a_place, ei_place or ou_place), where a_place is an additional category used to denote the place of articulation of 'a', ei_place for the place of articulation of 'e' or 'i', and ou_place for the place of articualtion of 'o' or 'u'.
   (c) Manner(plosive, nasal, lateral, lateral_ fricative, rolled, rolled_fricative, flapped, fricative, frictionless, a_ manner, ei_manner, or ou_manner) – where a_manner is an additional category used to denote the manner of articulation of a, etc.

   Data for the phoneme 'k' is stored as follows: vowel[K] = no; vocalised[K] = no; place[K] = velar; manner[K] = plosive.

   The category values for place and manner form an unordered set. When comparing the attributes of phonemes, no account is taken of the fact that the bilabial and labiodental positions are close together, while the glottal position is distant. Two bilablial phonemes are deemed to match with respect to that feature, while a bilabial and a labiodental are deemed not to match at all, just as a bilabial and a glottal do not match at all. By contrast, in Kessler's (1995) study of dialects of Gaelic, a measure called the 'phone-string comparison' was introduced. Using this graded measure, characters with a 'reasonable de-

gree of phonetic similarity' we considered more likely to be substituted for each other in related dialects. However, Kessler found that the use of a simple binary (yes/no) measure for matching phoneme features gave a more reliable estimation of the relative distances between dialects.

(3) A record is kept of corresponding 'weaker' and 'stronger' sounds which are encountered in lenition and fortition, using the data on p. 37 of Crowley. For example, the data item Fortition_lentition[B][P] = -1 denotes that 'b' is stronger than 'p'.

The types of sound change listed by Crowley, and the conditions required for their identification, are as follows:

(a) Forms of deletion and insertion:
Aphaeresis: deletion of the initial character from word 1;
Prothesis: insertion of the initial character into word 1;
Apocope: deletion of the final character from word 1;
Apocope2: insertion of the final character into word 1. Some operations, such as this one, are the reverse of an operation given by Crowley. These reverse operations were incorporated so that all regular sound changes can be identified, irrespective of which word list is presented first.

(b) Forms of substitution:
Fortition: if Fortition_lenition[character 1][character 2] = -1;
Lenition: if Fortition_lenition[character 1][character 2] = 1;
Vowel similarity: if both character 1 and character 2 are vowels, and the number of phonetic features (vocalisation, place and manner) in common is greater than a predetermined threshold (2).

(c) 2:1 operations:
Fusion: characters 1a and 1b (adjacent characters in word 1) fuse to become the single character 2 in the word of the other language if the number of phonetic features common to both (1a and 2) and (1b and 2) is greater than a predetermined threshold (1).
Vowel breaking2: the reverse of vowel breaking, i.e. two adjacent vowels in word 1 become just one in word 2 – the vowel in word 2 must be the same as one of the vowels in word 1.

(d) 1:2 operations:
Unpacking: character 1 in word 1 unpacks to become adjacent characters 2a and 2b in word 2, if the number of phonetic features common to both (1 and 2a) and (1 and 2b) is greater than a predetermined threshold (1).
Vowel breaking: a single vowel in word 1 becomes two adjacent vowels in word 2. The vowel in word 1 must be the same as one of the vowels in word 2.

(e) 2:2 operations:
Assimilation: the two adjacent characters in word 1 have more than a threshold (0) number of phonetic features in common, but the two corresponding adjacent characters in word 2 have more than a higher threshold (1) of features in common. Thus the two characters in word 2 have more phonetic features in common than those in word 1.
Dissimilation: the two adjacent characters in word 2 have more than a threshold (0) number of phonetic features in common, but the two corresponding adjacent characters in word 1 have more than a higher threshold (1) of features in common. Thus the two characters in word 1 have more phonetic features in common than those in word 2.

(f) 3:2 operations:
Syncope: deletion of a vowel from between two consonants. Thus characters (1a and 2a) and (1c and 2b) must be identical consonants, and character 1b must be a vowel.
Cluster Reduction: deletion of a consonant from between two consonants. Thus characters (1a and 2a) and (1c and 2b) must be identical consonants, and character 1b must be a consonant.

(g) 2:3 operations:
Epenthesis: insertion of a vowel between two consonants.

Excrescence: insertion of a vowel between two consonants.

The types of sound change described in this section are those which typically take place between a protolanguage and one of its daughter languages. Such changes will only be found between two daughter languages if the change has occurred in one of them (with respect to the protolanguage) but not the other.

EXPERIMENTAL RUN FOR THE DETECTION OF REGULAR SOUND CHANGES.

JAKARTA was used to align all pairs of cognate words at the character level in each of the four daughter language word lists. Examples of individual sound changes found can be seen in Table 3. In comaparing the Javanese 'awu' with the Malay 'abu' (meaning *ash*, as in the remains of fire), the 'a' and the 'u' of the 'awu' matched the 'a' and 'u' of 'abu' exactly. The replacement of 'w' by 'b' is an example of fortition. Since the total cost of this alignment is not more than 4, the word pair is assumed to be cognate, and the tally of occurrences of the sound changes a → a, w → b, and u → u are all incremented by 1. The word pair 'wataN' and 'perut' is deemed not to be cognate because the total cost of the alignment is over 4, so none of the operations required to transform one word into the other are counted towards the tally of sound changes found so far. The sound changes discovered in this way are collated, and each sound change occurring a threshold number of times (2) is deemed to be regular. The regular sound changes found for the Javanese-Malay comparison are shown in Table 4.

Table 3. Character Level Alignment for Javanese-Malay Word Pairs.

Javanese → Malay
[awu] [abu]
u → u match:
w → b fortition:
a → a match:
cost = 1
[wataN] [perut]
N → t substitution:
a → u substitution:
t → r substitution:
wa → pe dissimilation:
cost = 7
[anaq] [anak]
q → k fortition:
a → a match:
n → n match:
a → a match:
cost = 1
[tareq] [tarik]
q → k fortition:
e → i vowel_sim:
r → r match:
a → a match:
t → t match:
cost = 2

Table 4. Regular Sound Changes Found Between Javanese and Malay.

0	→ h	: 4
0	→ r	: 2
N	→ 0	: 2
N	→ N	: 11
a	→ a	: 36
b	→ b	: 8
c	→ c	: 2
d	→ j	: 2
e	→ e	: 4
e	→ i	: 12
g	→ g	: 4
h	→ h	: 5
i	→ i	: 12
k	→ k	: 9
l	→ l	: 10
m	→ m	: 6
n	→ n	: 14
o	→ 0	: 2
o	→ a	: 2
o	→ o	: 2
o	→ u	: 8
p	→ p	: 6
q	→ k	: 4
r	→ r	: 5
s	→ s	: 6
t	→ t	: 19
u	→ u	: 18
w	→ b	: 3
0	→ 0	: trivial

RECONSTRUCTION OF THE PROTOLANGUAGE PHONEMES

Program JAKARTA is used to identify the regular sound changes occurring for each language pair, i.e., Javanese-Madurese, Javanese-Malay, Javanese-Sundanese, Madurese-Malay, Madurese-Sundanese and Malay-Sundanese. The first task of program PRAGUE is then to read in these six tables of bilingual sound changes, and from these build up a table of regular four way sound correspondences. If the four daughter languages under study are referred to as L0, L1, L2 and L3, the technique is to look for instances where L0 and L1 have a regular change a → b, L0 and L2 have a regular change a → c, L0 and L3 have a regular change a → d, L1 and L2 have a regular change b → c, L1 and L3 have a regular change b → d and L2 and L3 have a regular change c → d. If all six conditions hold true, then L0, L1, L2 and L3 exhibit the regular sound change a → b → c → d.

The 'trivial' 0 → 0 (where 0 denotes the NULL symbol) change is routinely added to each list of bilingual sound correspondences, as it may be necessary to build up a four-way correspondence where a phoneme appears in only some of the four languages, and has no non-null equivalent in the others, as in the correspondence 0 = h = 0 = 0.

The lists of sound correspondences which hold true across all four languages are then used to derive the corresponding sounds in the protolanguage using four principles given by Crowley (1992, Chapter 5), as follows:

(1) Proposed reconstructions must involve sound changes that are plausible. Lenition is typically more plausible than fortition. Thus given the modern forms 'batu' and 'vatu' (Malay and Fijian for *stone*), it is more likely that the word in the protolanguage was 'batu', since this would mean that lenition had occurred in Fijian. If the protoform had been 'vatu', that would mean that the less likely process of fortition had occurred in Malay. Since the NULL phoneme is the weakest sound of all, the proposed protophoneme cannot be NULL.
(2) Look for sound correspondences that involve phonetically similar sounds.
(3) Reconstructions should involve as few changes as possible between the protolanguage and the daughter languages. If the corresponding sounds in four daughter languages are 't', 't', 't' and 'k', the corresponding sound in the protolanguage would probably be 't', since that would only involve one change with respect to the daughter language.
(4) A phoneme should not be proposed for the protolanguage if it is not found in any of the daughter languages.

These four principles are incorporated in a module which estimates the total degree of difference (cost) between each proposed phoneme (each phoneme in the alphabet is tried in turn) for the protolanguage with each of the corresponding phonemes in the daughter languages. A cost of 0 is given if the proposed protolanguage phoneme is identical with a daughter language phoneme, a cost of 1 is given if the protophoneme and the daughter phoneme differ in a manner favoured by principles (1) or (2), and a cost of 2 otherwise. For example, if we have the modern four-way sound correspondence t = t = t = k, and try 'k' as the protophoneme, the total cost will be 6. If we try 't', the cost will be 2, and if we try 'z', the cost will be 8. Thus 't' will be chosen as the most likely protophoneme.

A fifth principle regards conditioned (such as position dependent) sound changes, and this has been only partly implemented. Table 5 shows the protophonemes proposed by program PRAGUE for each four-way sound correspondence found for the set of daughter languages. Sometimes more than one of these four-way correspondences results in the same protophoneme being proposed, and these are distinguished by different numerals after each instance of the protophoneme. For example, the four-way comparison b = b = b = b suggests the protophoneme b1, while the comaparison w = b = b = b suggests the protophoneme b2. b1 and b2 may have differed in the protolanguage in one of two respects: they may have had similar, but not identical pronunciations (e.g., one was the phoneme

Table 5. Experimentally Found Corresponding Phonemes for Javanese, Madurese, Malay and Sundanese, and the Reconstructed Protophonemes.

Jav	= Mad	= Mal	= Sun	= PMJ
0	= h	= 0	= 0	= h1
N	= N	= N	= N	= N
a	= a	= a	= a	= a1
b	= b	= b	= b	= b1
c	= c	= c	= c	= c
e	= e	= e	= e	= e1
e	= e	= e	= i	= e2
e	= e	= i	= e	= e3
e	= e	= i	= i	= e4
g	= g	= g	= g	= g
i	= e	= i	= e	= i1
i	= e	= i	= i	= i2
k	= k	= k	= k	= k
l	= l	= l	= l	= l
m	= m	= m	= m	= m
n	= n	= n	= n	= n
o	= a	= a	= a	= a2
o	= o	= o	= o	= o1
o	= o	= u	= u	= o2
r	= r	= r	= r	= r
s	= s	= s	= s	= s
t	= t	= t	= t	= t
u	= o	= u	= u	= u1
u	= u	= u	= u	= u2
w	= b	= b	= b	= b2
0	= 0	= h	= h	= h2
0	= 0	= h	= 0	= h3
0	= 0	= 0	= q	= q
0	= 0	= 0	= 0	= 0

'b' while the other was a similar but distinct phoneme), or they may have been the same phoneme occurring in different positions. It is left to the intuition of the human linguist to decide which of these was the case. It is difficult to incorporate all possible contexts in which a phoneme might occur into a computer program because there are so many. The contexts considered by Crowley are listed above, while another, almost entirely different set is considered by Nothofer: initial, intervocalic, final, in the final closed syllable, before pause, when followed by u, before h, in the final syllable, before q, before i, between identical vowels, after u or o; after i or e; after u, and after i.

One other principle given by Crowley has not been implemented at all, namely that regarding the maintenance of balanced phonological systems.

RECONSTRUCTION OF THE PROTOLANGUAGE VOCABULARY

In the final part of program PRAGUE, the vocabulary lists for each daughter language are read in again, and each set of four words is aligned at the character level. Each phoneme in each daughter language word is then substituted for the corresponding phoneme in the protolanguage, if this is known. If these substitutions generate the same sequence of protolanguage phonemes for all four daughter languages, it can be assumed that the original word in the protolanguage has been plausibly reconstructed. If it is not possible to generate an identical sequence of protophonemes for all four languages, program PRAGUE tries to generate a sequence for just three of the languages, and failing that, just two. If it is not possible to generate an identical sequence of protophonemes for even two languages, the program reports that the word in the protolanguage could not be reconstructed for that meaning. Examples illustrating this process are shown in Table 6.

In comparing the daughter language words 'wulu' (Javanese), 'bulu' (Madurese), 'bulu' (Malay) and 'buluq' (Sundanese), all meaning *feather*, we see that the resulting character level alignment w = b = b = b is found in Table 5, and suggests the protophoneme b2. u = u = u = u yields u2, l = l = l = l yields l, u = u = u = u again yields u2 and 0 = 0 = 0 = q yields q. Thus the words in all four languages contribute to the word for *feather* in the protolanguage being reconstructed as 'b2 u2 l u2 q'.

In comparing the words 'mati', 'mate', 'mati' and 'maot' (all meaning *die*), the sequences m = m = m = m and a = a = a = a yield the protophonemes m and a respectively, but 0 = 0 = 0 = o is not found in Table 5. This means that these four characters cannot be substituted by an identical protophoneme, so it is not possible to reconstruct the word meaning *die* using all four languages. However, using only the first three languages gives the following reconstruction. There

Table 6. Sample Output for Program PRAGUE.

[wulu]	[bulu]	[bulu]	[buluq]
0	= 0	= 0	= q
u	= u	= u	= u
l	= l	= l	= l
u	= u	= u	= u
w	= b	= b	= b

PMJ = [b2 u2 l u2 q] from languages 0123

[mati]	[mate]	[mati]	[maot]
i	= e	= i	= 0
t	= t	= t	= t
0	= 0	= 0	= o
a	= a	= a	= a
m	= m	= m	= m

PMJ = [m a1 0 t i1|i2] from languages 012

[kabeh]	[kabbhi]	[semua]	[kabeh]
0	= i	= 0	= 0 =
h	= h	= a	= h =
e	= b	= u	= e =
b	= b	= m	= b =
a	= a	= e	= a =
k	= k	= s	= k =

PMJ = [k a1 b1 e1|e3 h 0] from languages = 03

[geteh]	[dara]	[darah]	[getih]
h	= 0	= h	= h =
e	= a	= a	= i =
t	= r	= r	= t =
e	= a	= a	= e =
g	= d	= d	= g =

PMJ = [d a1|a2 r a1|a2 h2|h3] from languages = 12
PMJ = [g e1|e3 t e2|e4 h] from languages = 03

[lan]	[ban]	[dan]	[jiN]
n	= n	= n	= N =
a	= a	= a	= i =
l	= b	= d	= j =

PMJ NOT FOUND

is only one entry in Table 5 where the first three columns are m = m = m, and that has the corresponding protophoneme m. a = a = a for the first three columns unambiguously yields the protophoneme a, while 0 = 0 = 0 yields 0 (the NULL character, which may be ignored) and t = t = t must yield t. However, the sequence i = e = i for the first three columns is found in two different rows, one giving the protophoneme i1 and one giving the protophoneme i2. This means that using just the first three languages, the protolanguage word for *die* is ambiguously reconstructed 'm a t i1' or 'm a t i2'.

In the example of 'kabeh', 'kabbhi', 'semua' and 'kabeh' (*all*), only the words in L0 and L3 were found to be cognate, although according to Nothofer's rules, the word in L1 would also be cognate. The example 'geteh', 'dara', 'darah', 'getih' (*blood*) yields two possible reconstructions, one cognate with 'geteh' and one with 'dara'. In such situations, linguists will look at languages closely related to, but outside the group of daughter languages under study. The Tagalog word for *blood* is 'dagat', suggesting that the earlier form should be cognate with 'dara', but this is not built into the program. No reconstruction was possible for 'lan', 'ban', 'dan' and 'jiN' (*and*).

RESULTS: ANALYSIS OF ERRORS

The output of program PRAGUE for each of the 195 sets of words in the word list was compared with the set of reconstructions given by Nothofer. In 55 cases, a PMJ form was reconstructed by both Nothofer and PRAGUE, but in only 13 cases were these reconstructions identical, such as 'mataq' (Nothofer) and 'mataq' (PRAGUE). Of the 42 non identical reflexes, 36 produced the same morpheme but differing pronunciations, as follows: In 9 cases there were differences in reconstructing the phonemes 'q', 'h' and 'k', which have NULL equivalents in some of the daughter languages, e.g., 'qanak' (Nothofer), 'anaq' (PRAGUE). In 9 cases the similar vowels 'e/i' and 'o/u' were reconstructed differently, e.g., 'kulit' and 'kulet'. In 4 cases JAKARTA failed to recognise that the Madurese compound sounds 'bh' and 'jh' should form 2:1 alignments with single consonants in the other languages, such regions being interpreted as separate 1:1 and 1:0 regions. This resulted in such differences in reconstruction as 'zauh' and 'jhauh'. In Nothofer's reconstructions, if Madurese has a double consonant where the other three languages have single consonants, the PMJ form is reconstructed with a double consonant (5 cases). However, using Crowley's third principle, PRAGUE assigns a single consonant in

such cases. For example, Nothofer gives 'bellah', while PRAGUE gives 'belah'. More complex phonemic differences in the reconstructions were found in 9 cases, e.g., 'tuhaq' (Nothofer) and 'tuwa' (PRAGUE). In 6 cases, the two methods produced a different morpheme altogether, such as 'qasuk' and 'qanj[eli]N'. In 32 cases Nothofer was able to produce a reconstruction but PRAGUE was not, while in 35 cases PRAGUE produced a reconstruction but Nothofer did not. For 74 word sets, neither Nothofer nor PRAGUE produced a reconstruction.

In this study, all four parts of the process of discovering the regular sound changes found between historically related languages described by previous authors have been implemented. In addition, program JAKARTA incorporates linguistically-motivated but language family independent preferences for sound substitution rules, and program PRAGUE enables the reconstruction of ancestor language (such as Proto-Malayo-Javanic) from its daughter languages alone. Although it was not possible to reproduce the results of a human linguist exactly, programs JAKARTA and PRAGUE can quickly produce outputs which will serve as a first estimate for the human historical linguist.

REFERENCES

Bloomfield, L. (1925). On the sound system of Central Algonquian. *Language, 1*, 130–156.

Covington, M.A. (1998). Alignment of multiple languages for historical comparison. *Proceedings of COLING 98, Vol. 1*, 275–280.

Crowley, T. (1992). *An introduction to historical linguistics*. Oxford University Press.

Damerau, F.J. (1975). Mechanization of cognate recognition in comparative linguistics. *Linguistics, 148*, 5–29.

Frantz, D.G. (1970). A PL/1 program to assist the comparative linguist. *Communications of the ACM, 13(6)*, 353–356.

Gale, W., & Church, K. (1993). A Program for aligning sentences in bilingual corpora. *Computational Linguistics, 19(1),* 75–102.

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

Kessler, B. (1995). Computational dialectology in Irish Gaelic. *7th Conference of the European Chapter of the Association for Computational Linguistics*. University College Dublin, 27-31 March, 1995.

Kruskal, J. (1983). An overview of sequence comparison. In D. Sankoff & J. Kruskal (Eds.), *Time warps, string edits and macromolecules: The theory and practice of sequence comparison* (pp. 1–44). Reading, Massachusetts: Addison-Wesley.

Lees, R. B. (1953). The basis of glottochronology. *Language, 29(2),* 113–127.

Lowe, J.B., & Mazaudon, M. (1994). The Reconstruction Engine: A computer implementation of the comparative method. *Computational Linguistics, 20*, 381-417.

McEnery, A.M., & Oakes, M.P. (1996). Sentence and word alignment in the CRATER project. In J. Thomas & M. Short (Eds.*), Using corpora for language research* (pp. 211–231). London: Longman.

Norman, J.M. (1975). *Elementary Dynamic Programming*. London: Edward Arnold.

Nothofer, B. (1975). *The reconstruction of Proto-Malayo-Javanic*. 's-Gravenhage: Martinus Nijhoff.

Simard, M., Foster, G., & Isabelle, P. (1992). Using cognates to align sentences in bilingual corpora. *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation (TMI 92).* Montreal, Canada, 67–81.

Swadesh, M. (1964). Linguistics as an instrument of prehistory. In D. Hymes (Ed.), *Language in Culture and Society: A Reader in Linguistics and Anthropology* (pp. 575–584). New York: Harper and Row.

Wagner, R.A., & Fischer, M.J. (1974). The string to string correction problem. *Journal of the ACM, 21*, 168.