Algorithms for Language Reconstruction

by

Grzegorz Kondrak

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Algorithms for Language Reconstruction

Grzegorz Kondrak

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2002

Genetically related languages originate from a common proto-language. In the absence of historical records, proto-languages have to be reconstructed from surviving cognates, that is words that existed in the proto-language and are still present in some form in its descendants. The language reconstruction methods have so far been largely based on informal and intuitive criteria. In this thesis, I present techniques and algorithms for performing various stages of the reconstruction process automatically.

The thesis is divided into three main parts that correspond to the principal steps of language reconstruction. The first part presents a new algorithm for the alignment of cognates, which is sufficiently general to align any two phonetic strings that exhibit some affinity. The second part introduces a method of identifying cognates directly from the vocabularies of related languages on the basis of phonetic and semantic similarity. The third part describes an approach to the determination of recurrent sound correspondences in bilingual wordlists by inducing models similar to those developed for statistical machine translation.

The proposed solutions are firmly grounded in computer science and incorporate recent advances in computational linguistics, articulatory phonetics, and bioinformatics. The applications of the new techniques are not limited to diachronic phonology, but extend to other areas of computational linguistics, such as machine translation.

Dedication

To Piotr Kondrak, My Uncle,

Who Started Me on My Way

Acknowledgements

- Graeme Hirst, my academic advisor, convinced me to get into the program and took the risk of allowing me to choose an unconventional research topic. He often knew better than me what was good for me.

- Members of my advisory committee, that is, Elan Dresher, Radford Neal, and Suzanne Stevenson, were always there when I needed help.

- Kevin Knight provided valuable comments about the contents of the thesis.

- The faculty in the Department of Linguistics, especially Elizabeth Cowper, Elan Dresher, Dianne Massam, and Keren Rice, made me feel like a member of their department.

- Hank Rogers got me hooked on phonetics. The drawings of the vocal tract included in this thesis are his.

- The members of the computational linguistics research group, including Melanie Baljko, Alex Budanitsky, Phil Edmonds, Neil Graham, Diana Inkpen, Eric Joanis, Daniel Marcu, and Gerald Penn, contributed to an exciting research environment.

- I thank my office-mates: Yiannis Papoutsakis for the existential discussions, Stuart Andrews for the music, and Iluju Kiringa for the tips.

- *Gemma, mi amor, mi vida, mi inspiración. Sin ti jamás habría terminado este trabajo.*

Contents

List of Tables

List of Figures

Chapter 1

Introduction

> *The goal of the computational linguist is to develop a computational theory of language, using the notions of algorithms and data structures from computer science.*
>
> James Allen, *Natural Language Understanding*

It is a common experience for second-language learners who struggle to memorize alien roots to unexpectedly encounter one that resembles a word from their native language. It is like seeing a familiar face in a crowd of strangers. At last, the form of a word gives a clue about its meaning.

How can the similarity of words across languages be explained? Sometimes the answer is easy. There can be no doubt that the English word *sushi* is a **borrowing** — the name was transferred from Japanese together with the food it denotes. Romance languages are full of **cognates**, such as Spanish *vida* and French *vie*, related words that have evolved over centuries from common Latin roots. How about Russian *nos* 'nose'? Common sense tells us that it must be cognate with English *nose* — such basic notions are rarely borrowed. Can such divergent languages as Russian and English originate from a common source? We can look the word up in an etymological dictionary, written by people who track word histories. But what if languages have no written records, as is the case with most of the New World languages? Is there a way to decide whether a pair of similar words are related? Can we guess what the **proto-form** of *nos* and *nose* was? Is it possible to reconstruct an entire **proto-language**?

Over the last two hundred years, historical linguists have developed methods for providing answers to such questions. By applying the **comparative method**, they have established beyond doubt that almost all European languages are related and belong to a single family. They have reconstructed the hypothetical Proto-Indo-European language and proved common origins of completely dissimilar words, such as Russian *sto* and English *hundred*. They have also analyzed and classified into families numerous languages from other continents.

As interesting as it seems, why would a computer scientist want to deal with such issues? What can she or he hope to contribute towards solving problems that had been studied for decades even before the computer was invented?

The language reconstruction task yields a number of problems that are easy to state but surprisingly hard to solve. Linguists have worked out solutions for many riddles related to particular language families. The question that a computer scientist wants answered is: "Is there an *algorithm* to do it?". In my opinion, the answer is no — the collection of heuristics called the comparative method of language reconstruction is not nearly sufficiently formalized to be called an algorithm.

Language reconstruction is an extremely time-consuming process that has yet to be accomplished for many language families. Greenberg [1993], in response to Ringe [1992], who criticized him for not applying the comparative method to the languages of the New World, estimates that the task would take longer than a lifetime. A computer system able to perform a fully automatic reconstruction of an unrecorded proto-language given wordlists representing two or more daughter languages would certainly be of great assistance to historical linguists. A successful implementation of just one or two steps of the comparative method on the computer could free the experts to divert their efforts to other, more challenging tasks.

The problems involved in language reconstruction may be classified as **driving problems**, a source of new methods and insights that are not restricted in application to historical linguistics. Like the basic research conducted in disciplines such as mathematics, the scope of applications may not be initially apparent or even predictable. Although the main focus of this thesis is diachronic phonology, I have striven to make the assumptions of the proposed solutions as general as possible, so that they may also be of use in other contexts.

Parallel bilingual corpora (**bitexts**) have been increasingly important in statistical natural language processing. Bitexts are collections of texts available in electronic form that are translations of each other. Since they convey nearly identical information by means of two different languages, bitexts are a rich source of translational equivalences between words and sentences. In order to exploit this encoded knowledge, it is necessary to align first the sentences within the corpus, and then the words within sentences. A number of researchers found that identifying words that are similar in both form and meaning can greatly facilitate the task of bitext alignment. A great majority of words in this class are cognates and borrowings. One of the goals of my research on the identification of cognates, which is discussed in this thesis, is to contribute to the improvement of the techniques used for bitext alignment as well as for the related task of inducing machine translation lexicons.

This thesis introduces new algorithms and methods for several stages of the language reconstruction process. The proposed solutions are language independent. I conducted evaluation experiments for the new methods, and did my best to objectively compare them with the previous proposals. I believe that simply implementing on a computer the traditional techniques that the linguists have been using for decades is not the right way to proceed. The intuitive criteria and broad knowledge that humans apply when dealing with such complex problems are often very difficult to encode in a computer program. My algorithms are based on some of the most recent research in computational linguistics, computer science, and bioinformatics. By attacking a much-studied problem from a different side, I hoped to obtain fresh insights and a better understanding of its nature.

The thesis is divided into three main parts that correspond to the principal steps of language reconstruction. Chapter 4 presents a new algorithm for the alignment of phonetic strings that combines a number of techniques developed for molecular biology with a scoring scheme for computing similarity on the basis of multivalued phonetic features. Chapter 5 introduces a method for identifying cognates in the vocabularies of related languages that employs the new alignment algorithm, as well as a procedure for estimating semantic similarity of words. Chapter 6 describes an approach to the determination of recurrent sound correspondences by inducing models similar to those developed for statistical machine translation.

Chapter 2

Background

Research in computational linguistics draws from both computer science and linguistics, and should be accessible to specialists from both disciplines. This chapter endeavours to clarify some of the concepts that recur throughout the thesis. Additional terms will be defined as they appear in the successive chapters.

2.1 Evaluating system effectiveness

The notions of precision and recall were developed in the field of information retrieval. **Precision** is the ratio of true positives to the sum of true positives and false positives, and **recall** is the ratio of true positives to the sum of true positives and false negatives. For example, an Internet search-engine query returns an ordered list of pointers, which are judged to be either relevant or not. In that context, true positives are the retrieved relevant documents, false positives are the retrieved non-relevant documents, and false negatives are the non-retrieved relevant documents. Suppose that the answer to our query about the names of Canadian provinces is "Alberta, Manitoba, Ontario, and Minnesota." Then, the precision of the answer is 0.75 (three out of four), and its recall is 0.3 (three out of ten).

In the context of cognate identification, precision is the proportion of selected pairs that are genuine cognates, and recall is the proportion of all cognates in the data that the system was able to identify. In general, there is a trade-off between improving increasing precision and increasing recall. We can achieve a 100% recall at the price of very low precision by assuming that all word pairs in the data are cognate. Conversely, we may be able to reach 100% precision by correctly guessing a single cognate pair.

In the situation where all word pairs are ranked according to their likelihood of being cognate, it is possible to compute precision and recall at any point in the ordered list. We may choose to compute precision at various levels of recall and then compute the average, which is called **uninterpolated average precision**. The **interpolated average precision** is similar, except for the assumption that precision can only decrease or stay at the same level as the recall level increases. In order to satisfy this constraint, if precision goes up while we are moving down the ordered list, the precision values for all lower recall levels are adjusted upward to match the current precision level.

2.2 Speech sounds

Phonetics is the study of speech sounds. The segmentation of continuous speech into a string of **phonetic segments** is not always a straightforward task. Phonetic segments are usually divided into two basic categories: **consonants**, which involve some type of obstruction in the vocal tract, and **vowels**, which are made with a very open vocal tract. Consonants are often classified according to the place of articulation (velar, dental, palatal, etc.), and according to the manner of articulation (stops, fricatives,

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

Table 2.1: Summary of English consonants.

nasals, etc.). Table 2.1 contains a summary of English consonants. Vowels have three basic articulatory qualities: **height**, **backness**, and **rounding**. For example, the vowel that occurs in the English word *rat* is identified as low, front, and unrounded. **Glides** are consonants that are phonetically similar to vowels. English has two glides: /y/ as in *yell*, and /w/ as in *well*.

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

Phonemes are phonetic segments that are distinctive in a given language. By contrasting with each other, phonemes serve to distinguish words. Phonetic segments that are distinct phonemes in one language are not necessarily contrastive in another language. The segments that are considered variants of a single phoneme are called **allophones**. For example, /t/ and /t^h/, which differ only in the degree of aspiration, are distinct phonemes in Thai, but no more than allophones in English.

Features are units of speech, smaller than a segment, describing an aspect of articulation. Features can be **unary**, **binary**, or **multivalued**, depending on the number of possible values. Every speech sound can be characterized by specifying a vector of feature values, but the characterization depends on the set of features that has been selected. A well known system of binary features is the one devised by Chomsky and Halle [1968]. A binary feature has exactly two possible values, a 'plus' value and a 'minus' value. For example, a binary feature specification for the bilabial voiced stop /b/ includes [+voice], [–coronal], and [–continuant]. Feature vectors of a number of sounds are given in Table 4.20 on page 44.

Morphology is the study of the internal structure of words. **Lexemes** are words in the sense of an abstract vocabulary item, which can have various realizations, or **word-forms**. For example, *choking*, *choke*, and *choked* are different realizations of the lexeme CHOKE. A word stripped of any inflectional affixes is called a **stem**. A **root** is the irreducible core of a word, which is always present in the realizations of the lexeme. The stem of the Latin infinitive *volāre* 'to fly' is obtained by discarding the inflectional ending *-re*, while its root is even more reduced: *vol-*.

2.3 Language change

All languages change through time. Table 2.2 gives an example of how much English has evolved within the last fourteen hundred years. Words that make up languages undergo **sound changes** (*nū* → *now*) as well as **semantic shifts** ('guardian' → 'ward'). **Lexical replacement** is a process in which lexemes drop out of usage altogether, and are substituted by other, unrelated words (*herigean* → *praise*). Morphological endings change and disappear as well (*-on* in *sculon*).

Old English:	Nū	sculon	herigean	heofonrīces	weard
Modern English:	Now	we should	praise	heaven-kingdom's	guardian

Table 2.2: The first verse of Caedmon's "Hymn" and its modern English translation.

When two groups of people that speak a common language lose contact with each other, their respective languages begin to diverge, and eventually become mutually unintelligible. In such cases, we may still be able to determine that the languages are **genetically related** by examining **cognates**, that is words that have developed from the same **etymon**, or **proto-form**. For example, French *lait*, Spanish *leche*, and Italian *latte* constitute a **cognate set**, as they are all descendants, or **reflexes**, of Latin *lacte*. In general, the longer the time that has passed since the linguistic split, the smaller the number of cognates that remain as a proof of a genetic relationship.

Because of gradual changes over long periods of time, cognates often acquire very different phonetic shapes. For example, English *hundred*, French *cent*, and Polish *sto* are all descendants of Proto-Indo-European **kmtom* (an asterisk denotes a reconstructed form). The semantic change can be no less dramatic; for example, English *guest* and Latin *hostis* 'enemy' are cognates even though their meanings are diametrically different. On the other hand, not all similar sounding words that have the same meaning are cognates. It can be a matter of chance resemblance, as in English *day* and Latin *die* 'day', or an instance of a **borrowing**, as in English *sprint* and Japanese *supurinto*. Borrowings, or **loan words**, are lexical items that have been incorporated (possibly in modified form) into one language from another.

An important phenomenon that allows us to distinguish between cognates and borrowings is the regularity of sound change. The regularity principle states that a change in pronunciation applies to sounds in a given phonological context across all words in the language. Regular sound changes tend to produce **regular correspondences** of phonemes in corresponding cognates. /d/:/t/ is a regular correspondence between English and German, as evidenced by cognate pairs such as *day* – *tag*, *dry* – *trocken*, and *drink* – *trinken*. Following Kessler [2001], I prefer to use the term **recurrent sound correspondences**, because in practice the matchings of phonemes in cognate pairs are more tendencies than hard-and-fast rules.

The **comparative method** of language reconstruction is the technique used by linguists to reconstruct proto-forms of the **parent language** by examining cognates in its **daughter languages**. It consists of several stages. After deciding that languages are related, words with similar meanings are placed side by side. Those pairs that exhibit some phonological similarity are identified as putative cognates. Next, the cognates are aligned by pairing related phonetic segments, and analyzed in order to find systematic correspondences. A proto-phoneme or a proto-allophone is posited for each established correspondence. The proto-forms that gave rise to the identified cognate sets are then reconstructed. The resulting phonological system of the proto-language is adjusted in order to conform to general linguistic principles. Naturally, the results of the subsequent steps can be used to refine the judgements made in the earlier ones.

The input data for establishing the relatedness of languages and reconstructing their common ancestor is often given in the form of a **bilingual wordlist**, a list of word pairs from two languages such that the corresponding words have the same, well-defined meaning. One of the most widely used set of meanings is the Swadesh 200-word list [Swadesh, 1952]. The 200 basic meanings in the list, given in full in Appendix A, are relatively resistant to lexical replacement and occur in most of the world's languages. The Swadesh 200-word lists have been compiled for a large number of languages. Table 2.3 contains an excerpt from the German–Albanian

		German	Albanian
1.	'all'	alə	ɟiθə
2.	'and'	unt	e
3.	'animal'	tīr	kafšə
4.	'ashes'	ašə	hi
5.	'at'	an	nə
6.	'back'	rükən	špinə
7.	'bad'	šlext	kec
8.	'bark'	rində	škəlbozə
9.	'because'	vayl	sepse
10.	'belly'	bawx	bark

Table 2.3: A bilingual wordlist [Kessler, 2001].

wordlist. Although the two languages are related, the entire list contains few cognates because after several thousand years of divergence, the lexical replacement process has obliterated almost all traces of the common origins.

A **vocabulary list**, or simply **vocabulary**, is a list of lexemes from a single language accompanied by **glosses** that explain their meaning. Glosses may be single words or complex phrases. I take a **vocabulary item** to mean a lexeme together with the corresponding gloss or glosses. A sample vocabulary list can be found on page 69.

Chapter 3

Related work

Computational diachronic phonology is concerned with two main tasks: deriving the modern forms from the old ones, and reconstructing the old forms from the modern ones. This chapter reviews previous algorithmic approaches to historical derivation and to comparative reconstruction. Other, closely related work is reviewed in subsequent chapters.

3.1 Historical derivation

Sound changes are regular in the sense that they normally apply to sounds in a given phonological context across all words in the language. Since the early seventies, there have been several proposals of derivation programs that take advantage of this regularity in order to simulate evolution of languages. While most of the programs discussed in this section have only a modest algorithmic content, they constitute starting points for more-computational approaches described in other sections.

The derivation programs can be used to verify the correctness of a particular set of sound-change rules and their relative chronology. If there are discrepancies between the expected and the actual output, it may be possible to refine the set of rules in an interactive way. They may also be of assistance in the construction of etymological dictionaries, or serve as a demonstration of the historical development of a language for educational purposes.

Eastlack [1977] provides a typical example of a derivation program. He implemented what he calls a computer simulation of systematic sound change in Ibero-Romance. The program contains about 90 rules that link first-century Latin with twelfth-century Castilian Spanish. Each sound-change rule is written as a separate module. The output consists of derivations that include not only the initial and final forms but also all the intermediate ones, together with information identifying the rules that triggered the changes. One apparent drawback is that syllable boundaries must be marked directly in the input, although it is not difficult to detect them automatically.

Burton-Hunter [1976] implemented the evolution of Latin word-forms into their Old French counterparts through the intermediate stage of Vulgar Latin. She chose Latin because it is a relatively well-documented proto-language, and French because it had undergone the greatest changes compared to other Romance languages. The long-term goal was to build a system consisting of a database of cognates for all Indo-European languages and programs that would perform both derivations and reconstructions.

Another researcher who used Latin as the proto-language was **Hartman** [1981], who implemented a program simulating the evolution of Spanish. His main innovation is representing phonemes as binary vectors in which every bit stands for a single binary feature. This allows the implementation of sound changes in a way that closely mirrors their phonological formulations. However, since there is no universally accepted set of features, their selection must necessarily be arbitrary.

An early paper by **Smith** [1969] provides an example of a derivation program that deals with a much larger time distance. The goal was to derive modern Russian from reconstructed Proto-Indo-European forms. These two languages are separated by at least 5000 years of mostly undocumented changes, as compared to about 1500 years that have passed since Latin split into Romance vernaculars. Out of the total of 650 Proto-Indo-European etyma that were examined, almost 90% have left no reflexes in Russian, and so could not be used to verify the correctness of the program. Of the remaining 69 etyma, the generated form exactly matched the corresponding Russian word in only 9 cases, which prompted Smith to rather gloomily conclude that "historical linguistics may have grossly overestimated the exceptionless character of sound change." The poor performance of the program, however, was most likely also due to the incompleteness of the implemented set of sound changes. Appendix B contains some results of my implementation for modeling the phonological evolution of Polish that contradict Smith's conclusion.

Remmel [1980] reports a computer procedure for modeling the evolution of vowels in Estonian. What is different about his approach is that he uses acoustic formants for the description of sound changes. Formants are concentrations of energy at specific frequencies of a sound wave. For example, for the vowel that occurs in the English word *pet*, the values of the first three formants are 700 Hz, 2100 Hz, and 3100 Hz, respectively. Every phoneme represents thus a point in an n-dimensional space, where n is the number of formants. A sound change can be represented as a vector or a curve in the n-dimensional space, which makes it easy to express "non-discrete" changes that take place over a period of time. Unfortunately, the paper provides few details of how this approach may work in practice.

Raman *et al.* [1997] use derivation programs to develop distance measures between parent and daughter languages. The diachronic relationship between languages is modeled as a Probabilistic Finite State Automaton, which represents the phonological complexity of the derivation process. The length of the minimal description of the automaton is taken as a measure of the distance between the earlier and the later form of a language. The authors apply the model to the modern Chinese dialects in order to establish their relative distance from the common proto-language.

3.2 Comparative reconstruction

The programs discussed in this section attempt to derive the proto-forms from the modern forms on the basis of regular sound correspondences provided by the user.

Hewson [1974] produced a dictionary of the Proto-Algonquian from wordlists representing four contemporary Algonquian languages. He does not claim to have reconstructed the proto-language – its sound system and all sound correspondences had already been established by other historical linguists. Rather, his contribution was to increase the number of reconstructed words from a few hundred to over four thousand.

The reconstruction process can be outlined as follows. First, from each input word of the daughter languages, every possible proto-form was generated using the provided set of regular sound correspondences. The resulting list was sorted alphabetically in order to detect the words from different languages that had identical proto-projections. The potential cognates were subsequently analyzed by a linguist whose job was to determine whether they are in fact reflexes of the same proto-form and to decide on its exact form.

Automatic processing of the linguistic data led to a tremendous reduction of time necessary to devote to the task. Hewson claims that with the aid of computer, the linguists were able to reconstruct about 250 items within a few hours. The final dictionary [Hewson, 1993] contains over 4000 reconstructed proto-forms incorporating the evidence of over 12,000 modern forms.

Hewson [1989] points out that once there is a sufficiently large database of forms and reconstructions, it is relatively easy to produce "spin-off" applications. In the case of the Proto-Algonquian dictionary, the applications that proved useful for related research included a concordance of consonant clusters and a dictionary of word-formatives.

A similar project is reported by **Johnson** [1985]. The goal was also to produce a comparative dictionary of the Yuman family of languages spoken in the Southwest United States. A relatively modest programming effort resulted in a substantial time saving over manual compilation of the dictionary. Standard tools such as **awk** and **sort** were used in preference to specialized programs. The reconstruction of proto-forms was performed by a simple substitution of phonemes according to the sound correspondences provided by linguists. Naturally, such a procedure works only for some sound changes. Nevertheless, Johnson reports that "the number of cases where two related roots do not reconstruct to the same form is fairly low."

The *Reconstruction Engine* of **Lowe and Mazaudon** [1994] is an impressive set of programs designed to aid the historical linguist in reconstruction work. By processing complete lexicons of modern languages, they can establish cognate sets together with reconstructions; at the same time, they can generate reflexes from the provided proto-forms. The user is allowed a high degree of flexibility in choosing the transcription system, the method of data organization, and the type of constituents used in the analysis. The inclusion of such features shows that the *Reconstruction Engine* was designed with a practical utility in mind. However, the authors acknowledge that in order to be useful to other researchers in historical linguistics, their programs would have to be integrated with a larger package containing an appropriate interface. Moreover, since it was developed and tested on a single family of languages (Tamang), it remains to be seen if it can be applied to other language families with equal success. At the time of writing, the *Reconstruction Engine* has yet to be made publicly available.

The common characteristic of the three approaches discussed thus far is that they attempt to meet the challenge of proceeding backwards in time by providing their programs with previously determined regular sound correspondences. Unfortunately, the determination of sound correspondences is one of the most challenging steps of the reconstruction process. A linguist whose job is to retrace the development of a language family may have only wordlists of modern forms at her disposal. In all but a handful of cases, there are no records that demonstrate the form of the proto-language. Most of the indigenous American languages, for instance, have no historical records at all.

Another difficulty is caused by the fact that a table of correspondences is not powerful enough to capture a series of phonological changes. In contrast with the historical derivation, proceeding in the opposite time direction is not, in general, a deterministic process. If a later change obliterates the environment that conditioned an earlier change, or if two independent phonemes merge into one, the reverse derivation of forms is all but impossible. The implicit assumption that all sound changes operate independently, present in the methods employing tables of correspondences, is untenable.

3.3 Comparative reconstruction from wordlists

The problem of automatically reconstructing a proto-language from wordlists that represent its descendants is extremely challenging. I have found only a few publications describing programs designed for this purpose, and none of them is close to providing a functional solution.

Kay [1964] represents an interesting attempt to formalize a large part of the comparative method in terms of propositional logic. The criterion that guides the search for correspondences is the minimization of the number of proto-phonemes necessary to account for the input data. The identified correspondences determine which word pairs in a bilingual list are cognate, and how they should be aligned. The author stops short, however, of converting the list of correspondences into proto-phonemes.

Kay develops his model with impressive rigor and generality. At one point in his paper, the entire problem of language reconstruction seems to be reduced to finding a satisfying assignment to a single propositional formula. Unfortunately, his solution is computationally impractical. According to my calculations, the number of elementary propositions generated for a pair of words of length n and m is $\sum_{k=1}^{n} k \binom{n-1}{k-1}\binom{m-1}{k-1}$, which means that a formula corresponding to two lexemes made of sixteen segments would contain over a billion terms, even *before* its conversion to disjunctive form. The program that implemented the model could successfully handle only trivially small data sets (the English-German wordlist given in the paper contains four pairs). The ability of the model to handle noisy data is also doubtful.

Remmel [1979] claims to have implemented a complete system for performing all stages of linguistic reconstruction. He even includes a sample set of seven wordlists and a corresponding list of 30 reconstructed words. According to him, most of the generated forms are identical with the commonly accepted reconstructions (which are not, however, included in the paper). There are virtually no details of the design; the description of the whole system takes less than 300 words.

Oakes [2000] describes a set of programs that together perform several steps of the comparative method, from the determination of regular correspondences in wordlists to the actual reconstruction of the proto-forms. The paper contains experimental results of running the programs on a set of wordlists representing four Indonesian languages, and compares those to the reconstructions found in the linguistic literature. I consider Oakes's implementation as important because his goal of performing fully automatic reconstruction from unprocessed wordlists is very similar to what I hope to achieve in the future. However, his approach is very different from the one advocated in this thesis. Rather than creating a program for reconstructing a *specific* proto-language, my goal is to develop methods that are applicable to *any* language family. The author has graciously provided me with the source code of his programs. I will discuss various components of his system in the following chapters.

Chapter 4

Phonetic alignment

Identification of the corresponding segments in phonetic strings is a necessary step in many applications in both diachronic and synchronic phonology. Usually we are interested in aligning strings that represent forms that are related in some way: a pair of cognates, or the underlying and the surface forms of a word, or the intended and the actual pronunciations of a word. Table 4.1 shows an example alignment of two cognates, Spanish *poner* and Italian *porre*, which originate from Latin *ponere* 'to put'. 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.

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

Table 4.1: An example alignment of two cognates.

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 dynamic programming algorithm for string alignment [Wagner and Fischer, 1974], although other algorithms can be used as well. (Dynamic programming is a technique of efficiently solving problems by combining previously computed solutions to smaller subproblems.) The task of finding the optimal alignment is closely linked to the task of calculating the distance between two strings. The basic dynamic programming algorithm accomplishes both tasks. Depending on the application, either of the results, or both, can be used.

Within the last few years, several different approaches to phonetic alignment have been reported. Covington [1996] used depth-first search and a special distance function to align words for historical comparison. Somers [1998] proposed a special algorithm for aligning children's articulation data with the adult model. Gildea and Jurafsky [1996] applied the dynamic programming algorithm to pre-align input and output phonetic strings in order to improve the performance of their transducer induction system. Kessler [1995] and Nerbonne and Heeringa [1997] adopted a similar approach in order to estimate phonetic distance between dialects of Irish and Dutch, respectively. All the algorithms are described in more detail in Section 4.2.

In this chapter, I present a new algorithm for the alignment of cognates. It combines various techniques developed for sequence comparison with a scoring scheme for computing phonetic similarity on the basis of multivalued features. The new algorithm performs better, in terms of accuracy and efficiency, than comparable algorithms reported by Covington [1996], Somers [1999], and Oakes [2000]. The algorithm is applicable not only to diachronic phonology but also to any other contexts in which it is necessary to align phonetic strings, including the applications identified in the preceding paragraph.

4.1 Sequence comparison

Sequence comparison is a term used to describe a number of techniques and algorithms for comparing two or more sequences of elements drawn from some alphabet. The algorithms have been successfully applied to problems in such diverse fields as molecular biology, speech recognition and gas chromatography [Kruskal, 1983]. The increased interest in the analysis and processing of DNA sequences resulted in the rapid development of related algorithms. It is not surprising that the techniques developed for molecular biology are often applicable in diachronic phonology. In both cases, sequences made of a limited set of units undergo evolutionary changes and splits.

In order to avoid terminological confusion, I use the word **string**, as commonly used in the computer science literature, rather than **sequence**, which is preferred in the context of biological applications. **Phonetic strings** are either entire words, or their parts, such as roots or stems. A **substring** is *not* the same thing as a **subsequence**. The former must be composed of contiguous units, while the latter need not.

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

In general, the number of possible alignments grows exponentially with the length

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

Table 4.2: The metric axioms.

of strings. Usually, we are interested in finding the alignment that minimizes a certain numerical function called *edit distance*. For example, we can define the distance as the minimum number of substitutions and indels necessary to convert one string into another (the so-called Levenshtein distance). A distance function that satisfies the axioms in Table 4.2 is called a *metric*.

The distance function can be made more complex by conditioning the cost of substitution and/or indels on the type of units involved. The distance between two strings is then the minimum sum of the costs of all edit operations necessary to convert one string into another.

Recently there has been a tendency in molecular biology to move away from edit distance toward probabilistic models, such as Hidden Markov Models [Durbin *et al.*, 1998]. It is not clear at this point whether HMMs indeed outperform standard alignment methods. While HMMs are not dealt with in this thesis, the comparison of the two paradigms in the context of of phonetic alignment is a fascinating future research topic.

4.1.1 The basic dynamic programming algorithm

The basic dynamic programming algorithm [Wagner and Fischer, 1974] accomplishes two related tasks: finding the optimal alignment of two strings, and calculating the distance between them. Given two strings of length n and m, the algorithm requires $O(nm)$ time to calculate the minimal edit distance plus $O(n+m)$ time to determine the corresponding alignment. Since I refer to the algorithm many times throughout this thesis, the time invested in studying the pseudocode given in Tables 4.3 and 4.4 will certainly pay off later.

Table 4.3 presents the first part of the algorithm, which computes the minimum distance between two strings. The distance is calculated recursively for successively larger substrings. $D[i,j]$ holds the minimal distance between the initial substrings of a and b containing i and j elements, respectively. $\delta(x,y)$ is the substitution cost between elements x and y. Similarly, $\delta(-,y)$ and $\delta(x,-)$ denote the insertion and deletion cost, respectively. Initially, the first row and the first column of D are filled by the multiples of the indel cost (lines 1.1–1.5). The main loop (lines 1.6–1.10) calculates the value of $D[i,j]$ by taking the minimum of three distances, which correspond to three possible ways of extending a partial alignment: by adding a substitution (line 1.8), deletion (line 1.9), or insertion (line 1.10). At the end of the computation, the value of $D[n,m]$ is the actual distance between the complete strings (line 1.11).

Table 4.4 presents the second part of the algorithm. The optimal alignment is retrieved from D by starting at $D[n,m]$ (line 2.1) and tracing back through the entries until the entry $D[0,0]$ is reached (line 2.2). The next entry of D after the current one depends on the choice that was made in the first part of the algorithm: conditions in line 2.3, 2.6, and 2.9 correspond to lines 1.9, 1.10, and 1.8, respectively. In fact, the choices made in part 1 can be remembered in a separate $n \times m$ array,

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

Table 4.3: Computing the distance between two strings.

```
i := n, j := m
while i ≠ 0 or j ≠ 0 do
    if D[i, j] = D[i − 1, j] + δ(a_i, −) then
        print ((a_i, −))
        i := i − 1
    else if D[i, j] = D[i, j − 1] + δ(−, b_j) then
        print ((−, b_j))
        j := j − 1
    else
        print ((a_i, b_j))
        i := i − 1, j := j − 1
```

Table 4.4: Retrieving the optimal alignment.

$D(i,j)$			c	o	u	l	e	u	r
		0	1	2	3	4	5	6	7
	0	**0**	1	2	3	4	5	6	7
c	1	1	**0**	1	2	3	4	5	6
o	2	2	1	**0**	**1**	2	3	4	5
l	3	3	2	1	2	**1**	2	3	4
o	4	4	3	2	3	**2**	**3**	4	5
u	5	5	4	3	2	3	4	**3**	4
r	6	6	5	4	3	4	5	4	**3**

Table 4.5: The array D after the computation of the distance between two strings.

thus eliminating the need to redo the summations in lines 2.3 and 2.6.

Table 4.5 shows a completely filled array D after computation of the alignment between *colour* and *couleur*. The insertion/deletion cost is assumed to be 1. The substitution cost is assumed to be 0 for identical segments, and 2 otherwise. The values shown in boldface indicate the retrieved path that corresponds to the alignment shown in Table 4.6.

c	o	u	l	-	e	u	r
c	o	-	l	o	-	u	r

Table 4.6: An alignment retrieved from the array shown in Table 4.5.

4.2 Previous alignment algorithms

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

Covington [1996] developed an algorithm for the alignment of cognates on the basis of phonetic similarity. In a follow-up paper [1998], he extended the algorithm to align words from more than two languages. His algorithm consists of a specially designed evaluation metric and a 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 indel cost (Table 4.18 on page 42). The values of the metric range from 0 for two identical consonants to 100 for two segments with no discernible similarity. 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 Section 4.5.

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]. The author observes that "alignment can become somewhat arbitrary" in the cases where the compared strings are not very similar. After running 'CAT' on Covington's test data, he concludes that, in terms of accuracy, it is as good as Covington's algorithm. I point out a weakness in Somers's approach in Section 4.3.1.

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. The pre-alignment is performed using the dynamic programming algorithm and a metric that is based on 26 binary features. The cost of substitutions is a straightforward Hamming distance between two feature vectors. The cost of indels is set at one quarter of the maximum possible substitution cost. Because the surface forms are generated directly from the underlying forms by the application of a few simple phonological rules, the alignment algorithm need not be sophisticated.

Nerbonne and Heeringa [1996; 1997; 1999] investigate the problem of measuring phonetic distance between dialects. The distance between two dialects is estimated by taking the sum of Levenshtein distances between two sets of corresponding words. They experimented with at least two different feature systems, but they don't provide the details of the systems. The cost of indels is set at half the average of all substitutions. The individual distances are calculated using the dynamic programming algorithm. The computed distance is normalized by dividing its value by the length of the longer word. Realizing that "not all features are equally important in classifying the sounds", the authors tried weighting each feature by information gain, which was computed as the average entropy reduction a feature represents when known. However, they found that the weighting had an adverse effect on the quality of the alignments. They also concluded that the Manhattan distance is preferable to both Euclidean distance and Pearson correlation.

Kessler [1995] was also interested in grouping dialects. The dialects were represented by wordlists, each containing about 50 concepts. He tested several different approaches for computing distance between dialects. The most sophisticated method employed twelve multivalued phonetic features. The numeric feature values were assigned arbitrarily, and all features were given the same weight. The distance between phonetic segments was calculated as the difference averaged across all twelve features. The cost of indels is not specified in the paper. Kessler found that the feature-based method performed worse than a simpler phoneme-based method, which employed a binary identity function between phonemes. The dynamic programming algorithm was used for computing the minimal distance between words in both methods.

Oakes's [2000] program JAKARTA contains a phonetically-based alignment algorithm, whose ultimate purpose is the discovery of regular sound changes. The array of edit operations, which are adapted from a historical linguistics textbook, is impressive. It includes lenition, fortition, aphaeresis, apocope, syncope, cluster reduction, excresescence, dissimilation, epenthesis, prothesis, fusion, vowel breaking, assimilation, and dissimilation. The conditions that must be satisfied to trigger the operations are also implemented. 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. In addition, distinct phonetic segments can have identical feature assignments. The distance function is therefore somewhat coarse.

Some properties of the approaches described in this section are juxtaposed in Table 4.7. The label *explicit* identifies the intended function of the algorithm, while the label *implicit* marks the functionality that is present but not overtly used.

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

Table 4.7: Comparison of phonetic alignment/distance algorithms.

4.3 Finding the optimal phonetic alignment

The dynamic programming algorithm 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.

4.3.1 Greedy 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 4.8). 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

t	e	w	o	s
d	i	v	u	t

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

Table 4.8: A correct and an incorrect alignment of a hypothetical cognate pair.

never backtracks is not guaranteed to find the optimal alignment.

Somers [2000] points out 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 diachronic phonology, where stress is too volatile to depend on. Even closely related languages may have different stress rules. For example, stress regularly falls on the initial syllable in Czech and on the penultimate syllable in Polish, while in Russian it can fall anywhere in the word. Somers [1999] nevertheless attempts to apply his algorithm to the alignment of cognates. In Section 4.8, I will examine the alignments reported in that paper.

4.3.2 Tree 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. Why does Covington prefer to use such a costly procedure even though he is aware that a more efficient algorithm exists?

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

> First, the strings being aligned are relatively short, so the efficiency of dynamic programming on long strings is not needed. Second, dynamic programming normally gives only one alignment for each pair of strings,

> but comparative reconstruction may need the n best alternatives, or all that meet some criterion. Third, the tree search algorithm lends itself to modification[1] 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 vocabularies of Algonquian languages [Hewson, 1999] 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 certain to cause a painful slow-down. Moreover, combining the alignment algorithm with some sort of strategy for identifying cognates on the basis of phonetic similarity is likely to require comparing thousands of words against one another. Having a polynomially bound algorithm in the core of such a system is crucial. In any case, since the 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 Section 4.4, I demonstrate that it is possible to retrieve from the distance matrix 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.

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

4.4 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. Appendix C contains the code that incorporates most of the extensions into a phonetic alignment algorithm.

4.4.1 Retrieving a set of best alignments

The algorithm in Table 4.4 constructs a single best alignment on the basis of the matrix D. At times, it may be desirable to find a number of alternative alignments that are close to the optimum. The basic dynamic programming algorithm can be adapted to such a task.

Myers [1995] describes a straightforward modification of the basic dynamic programming algorithm that produces all alignments that are within ϵ of the optimal distance. The idea is to recursively recover only those alignments that correspond to distances below the threshold score of $D[n,m]+\epsilon$. Let $T(i,j)$ be the distance accumulated during the traversal from $D[n,m]$ to $D[i,j]$. On reaching $D[i,j]$, the three conditions shown in Table 4.9 are checked, and a distinct alignment is constructed for each satisfied condition. This is in contrast with the algorithm in Table 4.4, which maintains a unique alignment throughout the computation. Appendix C contains actual C++ code that integrates this extension into the alignment algorithm.

In order to find the k-best alignments, the matrix D can be viewed as a graph with nodes corresponding to the entries in the D matrix, the source at $D[n,m]$, the sink at $D[0,0]$, and the arc lengths set according to the δ function. A number of polynomial-

3.3	$D[n,m] + \epsilon \geq T(i,j) + D[i-1,j] + \delta(a_i, -)$
3.6	$D[n,m] + \epsilon \geq T(i,j) + D[i,j-1] + \delta(-, b_j)$
3.9	$D[n,m] + \epsilon \geq T(i,j) + D[i-1,j-1] + \delta(a_i, b_j)$

Table 4.9: The conditions for finding near-optimal alignments.

time algorithms for calculating the k-shortest paths from source to sink in order of length are available from the operations research literature [Fox, 1973]. 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.

4.4.2 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 dynamic programming algorithm of Table 4.3 can be adapted to compute the similarity by simply replacing the *min* in line 1.8 by *max*. Naturally, the function δ will denote the similarity score rather than the edit distance.

The similarity approach is clearly 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 in the following section.

4.4.3 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 correct alignment of Cree *āpakosīs* and Fox *wāpikonōha* 'mouse' (Table 4.10) matches the roots of the words and leaves out the unrelated affixes. (Double bars delimit the aligned substrings.) Such an affix-stripping behaviour is much more difficult to achieve with global alignment.

	‖	ā	p	a	k	o	‖	sīs
w	‖	ā	p	i	k	o	‖	nōha

Table 4.10: An example of local alignment.

It should be clear by now 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.

The basic dynamic programming algorithm can easily be modified to perform local alignment (Table 4.11). The interpretation of $S[i,j]$ is changed to the maximal similarity between *suffixes* of the initial substrings of a and b containing i and j elements, respectively. The first row and the first column of S are initialized with zeroes. $S[i,j]$ is set to the minimum of four rather than three values, the fourth being zero. This has an effect that no entry of S is a negative number. A zero entry is

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

Table 4.11: The basic algorithm for computing local similarities between two strings.

interpreted as the alignment between the empty suffixes of the initial substrings. The optimal similarity is no longer found in $S[n, m]$, but rather it is equal to the maximum entry in S. The optimal alignment is retrieved by tracing back through the array, starting at the maximum entry but stopping as soon as a zero entry is found.

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 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, as in the alignment in Table 4.12. 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.

	‖	ā	p	a	k	o	s	ī	s	‖	
w	‖	ā	p	i	k	o	-	-	-	‖	nōha

Table 4.12: An example of semiglobal alignment.

‖	-	ā	p	a	k	o	‖	sīs
‖	w	ā	p	i	k	o	‖	nōha

Table 4.13: An example of half-local alignment.

The modifications required to make the dynamic programming algorithm perform semiglobal alignment are straightforward. As in the case of local alignment, the first row and column of S have to be initialized with zeroes. However, the zero option (line 11 in Table 4.11) is not added. The optimal similarity is the maximum entry in the union of the last row and the last column of S. Starting from this entry, we can retrieve the optimal alignment using the usual method, stopping when we reach one of the zeroes in the first row or column.

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, as exemplified in Table 4.13.

One way to implement half-local alignment is to fully penalize the leading indels, but make the trailing ones free. The opening phase of the alignment is performed as in the *global* case, while its completion obeys the *local* rules. The particulars of this approach can be gleaned from the code in Appendix C.

4.4.4 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

5.8	$AG[i,j] = max(AG[i-1,j] - s,\ S[i-1,j] - (r+s))$
5.9	$BG[i,j] = max(BG[i,j-1] - s,\ S[i,j-1] - (r+s))$
5.10	$S[i,j] = max(S[i-1,j-1] + \delta(a_i,b_j),\ AG[i,j],\ BG[i,j])$

Table 4.14: The modification of the basic dynamic programming algorithm required to accommodate affine gap scores.

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. Two additional arrays (AG and BG) record the best alignment that ends with a gap in the corresponding string. $AG[i,j]$ is interpreted as the minimal distance between the initial substrings of a and b when the substring of a ends with a gap, Similarly, BG corresponds to the case when the substring of b ends with a gap. The body of the main loop in the modified algorithm is shown in Table 4.14 (lines 5.8–5.10 replace lines 1.8–1.10 in Table 4.3).

Gotoh is not totally categorical about the initial settings of AG and BG. I found that $AG[0,j]$ and $BG[i,0]$ must be set to $+\infty$ for all values of i and j. In this way, we can be sure that the distance between two unrelated strings of length 1 is in fact equal to $2(r+s)$, as required.

4.4.5 Additional edit operations

The basic set of edit operations consists of substitutions and insertions/deletions. Another useful operation is compression/expansion. In the compression/expansion operation two contiguous units of one string correspond to a single unit of the other string. The operations of compression and expansion are distinct in the context of

f	a	k	t
-	e	č	-

f	a	k	t
-	e	-	č

f	a	kt
-	e	č

Table 4.15: An example of cognate alignment that requires the operation of compression/expansion.

transforming one string into another by a series of edit operations. However, in the context of finding an optimal alignment, they are really mirror images of each other, in the same way as the operations of insertion and deletion. We could as well talk about various types of correspondences: $0:1$, $1:1$, $1:2$, etc.

In the context of the alignment of cognates, the compression/expansion operation facilitates the expression of complex correspondences. For example, in the alignment of stems of Latin *factum* and Spanish *hecho*, the affricate [č] should be linked with both [k] and [t] rather than with just one of them, because it originates from the merger of the two consonants. Therefore, the rightmost alignment in Table 4.15 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.

Oommen [1995] formally defines the string alignment algorithm that incorporates the compression/expansion operation. The modification is straightforward. The lines 4.8–4.11 of the algorithm given in Table 4.11 on page 36 should be substituted with the code shown in Table 4.16. Two additional cases are simply added to the *max* expression. Also, minor changes are required in the initialization of the first two rows/columns of matrix S, and in the retrieval of the optimal alignment from matrix S. Appendix C contains the details of the modifications.

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].

4.5 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. Table 4.17 shows the most elementary distance function defined on a subset of phonetic segments. It 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.

6.8	$S[i,j] := max($	$S[i-1,j-1] + \delta(a_i, b_j),$
6.9		$S[i-1,j] + \delta(a_i, -),$
6.10		$S[i,j-1] + \delta(-, b_j),$
6.11		$S[i-2,j-1] + \delta(a_{i-1}a_i, b_j),$
6.12		$S[i-1,j-2] + \delta(a_i, b_{j-1}b_j),$
6.13		$0\ )$

Table 4.16: The modification of the dynamic programming algorithm that incorporates the compression/expansion operation.

	a	i	y	n	p	r	s
a	0	1	1	1	1	1	1
i	1	0	1	1	1	1	1
y	1	1	0	1	1	1	1
n	1	1	1	0	1	1	1
p	1	1	1	1	0	1	1
r	1	1	1	1	1	0	1
s	1	1	1	1	1	1	0

Table 4.17: An elementary cost function.

4.5.1 Feature-based metrics

Covington [1996], for his cognate alignment algorithm, constructed a special distance function (Table 4.18). 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.

The rather cryptic explanation for assigning a lower penalty to "skips preceded by another skip" refers to the fact that in diachronic phonology not only individual segments but also entire morphemes and syllables are sometimes deleted in a single event. It is worth noting that the indel penalties in Covington's scheme can be expressed by an affine gap function (cf. section 4.4.4) with $r = 10$ and $s = 40$.

Contrary to its name, Covington's distance function is not a metric because it does not satisfy all of the required axioms enumerated in Table 4.2 on page 23. The zero property is not satisfied because the function's value for two identical vowels is

[2]In this context, *precede* is used to mean *immediately precede*.

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[2] by another skip in the same word (reflecting the fact that affixes tend to be contiguous)
50	Skip not preceded by another skip in the same word

Table 4.18: Covington's [1996] "evaluation metric".

greater than zero. Also, the triangle inequality does not hold in all cases; for example: $p(a, i) = 30$ and $p(i, y) = 10$, but $p(a, y) = 100$, where $p(x_1, x_2)$ is the penalty for aligning $[x_1]$ with $[x_2]$.

Table 4.19 illustrates the "low resolution" of Covington's distance function. 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$^{\mathrm{h}}$], and both *yacht* and *will* are treated identically as a glide-vowel-consonant string. Covington [1998] acknowledges that his distance function is "just a stand-in for a more sophisticated, perhaps feature-based, system".

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 entry stands for a single articulatory feature. Such a representation allows one to distinguish a large number of phonetic segments. The distance

	a	i	y	n	p	r	s
a	5	30	100	100	100	100	100
i	30	5	10	100	100	100	100
y	100	10	0	60	60	60	60
n	100	100	60	0	60	60	60
p	100	100	60	60	0	60	60
r	100	100	60	60	60	0	60
s	100	100	60	60	60	60	0

Table 4.19: A partial distance matrix for Covington's distance function.

between two segments is 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. Since neither Gildea and Jurafsky [1996] nor Nerbonne and Heeringa [1997] present their feature vectors in sufficient detail to perform the calculations, I adapted a fairly standard set of binary features from Hartman [1981]. Twenty-five letters of the Latin alphabet (all but q) were taken to represent a sample set of most frequent phonemes. The feature vectors corresponding to the set are shown in Table 4.20. Table 4.21 contains a portion of the corresponding distance matrix.

Table 4.22 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 name	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	t	u	v	w	x	y	z
[tense]	+	−	−	−	+	−	−	−	+	−	−	−	−	−	+	−	−	−	−	+	−	+	−	+	−
[spread glottis]	−	−	−	−	−	−	−	+	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
[voice]	+	+	−	+	+	−	+	−	+	+	−	+	+	+	+	−	+	−	−	+	+	+	−	+	+
[back]	+	−	−	−	−	−	+	+	−	−	+	−	−	−	+	−	−	−	−	+	−	+	+	−	−
[coronal]	−	−	+	+	−	−	−	−	−	+	−	+	−	+	−	−	+	+	+	−	−	−	−	−	+
[continuant]	+	−	−	−	+	+	−	+	+	−	−	−	−	−	+	−	+	+	−	+	+	+	+	+	+
[high]	−	−	+	−	−	−	+	−	+	+	+	−	−	−	−	−	−	−	−	+	−	+	+	+	−
[strident]	−	−	+	−	−	+	−	−	−	+	−	−	−	−	−	−	−	+	−	−	+	−	−	−	+
[round]	−	−	−	−	−	−	−	−	−	−	−	−	−	−	+	−	−	−	−	+	−	+	−	−	−
[syllabic]	+	−	−	−	+	−	−	−	+	−	−	−	−	−	+	−	−	−	−	+	−	−	−	−	−
[obstruent]	−	+	+	+	−	+	+	+	−	+	+	−	−	−	−	+	−	+	+	−	+	−	+	−	+
[nasal]	−	−	−	−	−	−	−	−	−	−	−	−	+	+	−	−	−	−	−	−	−	−	−	−	−
[consonantal]	−	+	+	+	−	+	+	+	−	+	+	+	+	+	−	+	+	+	+	−	+	−	+	−	+
[low]	+	−	−	−	−	−	−	+	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
[anterior]	−	+	+	+	−	+	−	−	−	+	−	+	+	+	−	+	+	+	+	−	+	−	−	−	+
[distributed]	+	+	+	+	+	−	+	+	+	−	+	−	+	−	+	+	−	+	+	+	−	+	+	+	+
[delayed release]	−	−	+	−	−	−	−	−	−	+	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−

Table 4.20: Feature vectors adopted from Hartman [1981].

	a	i	y	n	p	r	s
a	0	3	4	10	9	8	10
i	3	0	1	9	8	7	9
y	4	1	0	8	7	6	8
n	10	9	8	0	5	2	6
p	9	8	7	5	0	5	3
r	8	7	6	2	5	0	4
s	10	9	8	6	3	4	0

Table 4.21: A partial distance matrix based on binary features.

	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

Table 4.22: The clause-by-clause comparison of Covington's distance function and a feature-based distance function.

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 4.22 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.

4.5.2 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. [y], which is the initial sound of the word *you*, and [ǰ], which is the initial sound of the word *Jew*, have an astounding 9 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, as illustrated in Figure 4.1.

Ladefoged [1995] devised a phonetically-based multivalued feature system. This system was adapted by Connolly [1997] and implemented by Somers [1998]. It contains about twenty 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]. All feature values are assigned numerical values between 0 and 1.

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 concluded that it actually had a detrimental effect on the produced 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] vaguely proposes "using multivariate statistical techniques

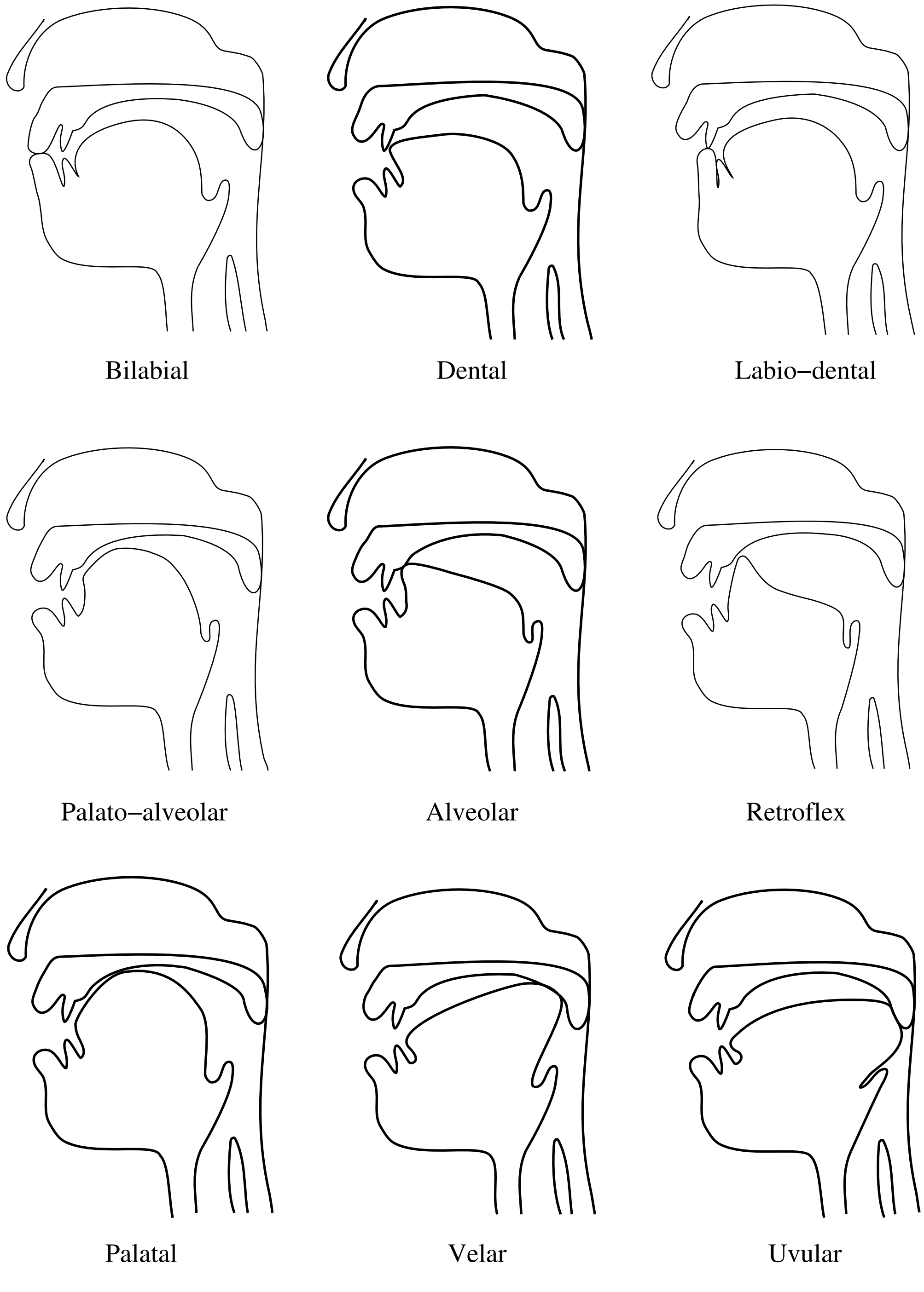

Figure 4.1: Places of articulation.

and a set of known 'good' alignments" for calculating the relative importance of each feature. In my opinion, it seems reasonable to try and derive the saliences automatically from a large corpus of aligned cognates (Durbin *et al.* [1998] describe methods of generating substitution matrices from alignment data). Unfortunately, it is not clear how to obtain a representative training set in the first place. Even if we ignore the problem of the bias introduced by selecting cognates from particular language pairs, the task of establishing the correct alignment of cognates is very time consuming and often requires detailed knowledge of the history of the languages involved.

4.5.3 Similarity and distance

Although all previously proposed alignment algorithms (Table 4.7) 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 identical 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 algorithm, below, I employ the similarity-based approach to comparing segments (cf. section 4.4.2). 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 4.4.3. 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.

It would be ideal to have, as Kessler [1995] puts it, "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 wealth of data available for DNA sequences is simply not attainable in diachronic phonology. 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 matrix of such size by deriving each individual entry 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 multivalued features with salience coefficients seems to be the best solution.

4.6 The algorithm

Many of the ideas discussed in previous sections have been incorporated into the new algorithm for the alignment of phonetic strings. 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.

Table 4.23 contains the main components of the algorithm. First, the dynamic programming approach is applied to compute the similarity matrix S using the σ scoring functions. For simplicity, $S(i, j)$ is defined to return $-\infty$ if either i or j is negative. Since local alignment is assumed, the first row and the first column of the matrix S are initialized with zeros. The optimal score is the maximum entry in the whole matrix. Every matrix entry that exceeds the threshold score T is the starting point of one or more alignments.

The recursive procedure **Retrieve** is shown in Table 4.25. The alignment is assembled by traversing the matrix until a zero entry is encountered. The parameters i and j are the coordinates of the current matrix entry. The partial alignment is stored on the stack *Out*, which is a last-in-first-out data structure. The parameter s accumulates the alignment's score. Note that the alignments are printed out in the order of decreasing indices. The order may be reversed by simply indexing the string starting from the end rather than from the beginning.

If a best alignment contains a sequence of two or more consecutive indels alternating between the two strings, some of the retrieved alignments may happen to be equivalent. In such a case, a unique alignment can be obtained by filtering out all alignments in which an indel in the first string is immediately followed by an indel in the second string. This constraint, which corresponds to the *ordered-alternating-skips* rule of Covington [1998], would suppress both the second and the third alignment of

algorithm Alignment
input: phonetic strings x and y
output: alignment of x and y
define $S(i, j) = -\infty$ when $i < 0 or j < 0$

for $i := 0$ **to** $|x|$ **do**
    $S(i, 0) := 0$
for $j := 0$ **to** $|y|$ **do**
    $S(0, j) := 0$
for $i := 1$ **to** $|x|$ **do**
    for $j := 1$ **to** $|y|$ **do**
        $S(i, j) := max($
            $S(i-1, j) + \sigma_{skip}(x_i),$
            $S(i, j-1) + \sigma_{skip}(y_j),$
            $S(i-1, j-1) + \sigma_{sub}(x_i, y_j),$
            $S(i-1, j-2) + \sigma_{exp}(x_i, y_{j-1}y_j),$
            $S(i-2, j-1) + \sigma_{exp}(x_{i-1}x_i, y_j),$
            $0)$

$T := (1 - \epsilon) \cdot \max_{i,j} S(i, j)$

for $i \leftarrow 1$ **to** $|x|$ **do**
    for $j \leftarrow 1$ **to** $|y|$ **do**
        if $S(i, j) \geq T$ **then**
            Retrieve$(i, j, 0)$

Table 4.23: The algorithm for computing the alignment of two phonetic strings.

g	ō	-		g	-	ō		-	g	ō
-	-	ī		-	ī	-		ī	-	-

Table 4.24: Three equivalent alignments.

English *go* and Latin *ire* 'go' shown in Table 4.24, but not the first one.

The scoring functions are defined in Table 4.26. C_{skip}, C_{sub}, and C_{exp} are the maximum scores for indels, substitutions, and expansions, respectively. C_{vwl} determines the relative weight of consonants and vowels. Phonetic segments are expressed as vectors of feature values. The function $\mathbf{diff}(p, q, f)$ returns the difference between segments p and q for a given feature f. Set R_V contains features relevant for comparing two vowels: *Syllabic*, *Nasal*, *Retroflex*, *High*, *Back*, *Round*, and *Long*. Set R_C contains features for comparing other segments: *Syllabic*, *Manner*, *Voice*, *Nasal*, *Retroflex*, *Lateral*, *Aspirated*, and *Place*. 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. The numerical values of features and their saliences are discussed in the next section.

4.7 Implementation

ALINE is a publicly available[3] implementation of the algorithm for the alignment of phonetic strings. The program is written in C++ and runs under Unix. 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. Although local comparison is the default,

[3] At `http://www.cs.toronto.edu/~kondrak/`.

```
procedure Retrieve(i, j, s)

if S(i, j) = 0 then
    print(Out)
    print("alignment score is s")
else
    if S(i-1, j-1) + σ_sub(x_i, y_j) + s >= T then
        push(Out, "align x_i with y_j")
        Retrieve(i-1, j-1, s + σ_sub(x_i, y_j))
        pop(Out)
    if S(i, j-1) + σ_skip(y_j) + s >= T then
        push(Out, "align null with y_j")
        Retrieve(i, j-1, s + σ_skip(y_j))
        pop(Out)
    if S(i-1, j-2) + σ_exp(x_i, y_{j-1}y_j) + s >= T then
        push(Out, "align x_i with y_{j-1}y_j")
        Retrieve(i-1, j-2, s + σ_exp(x_i, y_{j-1}y_j))
        pop(Out)
    if S(i-1, j) + σ_skip(x_i) + s >= T then
        push(Out, "align x_i with null")
        Retrieve(i-1, j, s + σ_skip(x_i))
        pop(Out)
    if S(i-2, j-1) + σ_exp(y_j, x_{i-1}x_i) + s >= T then
        push(Out, "align x_i x_{i-1} with y_j")
        Retrieve(i-2, j-1, s + σ_exp(y_j, x_{i-1}x_i))
        pop(Out)
```

Table 4.25: The procedure for retrieving alignments from the similarity matrix.

$$
\begin{aligned}
\sigma_{skip}(p) &= C_{skip} \\[1em]
\sigma_{sub}(p,q) &= C_{sub} - \delta(p,q) - V(p) - V(q) \\[1em]
\sigma_{exp}(p,q_1q_2) &= C_{exp} - \delta(p,q_1) - \delta(p,q_2) - \\
& \quad V(p) - max(V(q_1),V(q_2)) \\
\text{where} & \\
V(p) &= \begin{cases} 0 & \text{if } p \text{ is a consonant} \\ C_{vwl} & \text{otherwise} \end{cases} \\[2em]
\delta(p,q) &= \sum_{f \in R} \text{diff}(p,q,f) \times \text{salience}(f) \\
\text{where} & \\
R &= \begin{cases} R_C & \text{if } p \text{ or } q \text{ is a consonant} \\ R_V & \text{otherwise} \end{cases}
\end{aligned}
$$

Table 4.26: Scoring functions.

the program can be re-compiled to perform global and semiglobal alignment.

The behavior of the program is controlled by command-line parameters: the value of ϵ for producing near-optimal alignments, and the four constants described in the previous sections. The default values of the parameters are $\epsilon = 0$, $C_{skip} = -10$, $C_{sub} = 35$, $C_{exp} = 45$, and $C_{vwl} = 10$.

Table 4.27 enumerates the features that are currently used by ALINE and their salience settings. Feature values are encoded as floating-point numbers in the range $[0.0, 1.0]$. The numerical values of four principal features are listed in Table 4.28. The remaining features have exactly two possible values, 0.0 and 1.0. A part of the resulting similarity matrix is shown in Table 4.29.

The numerical values in Table 4.28 are taken from Ladefoged [1995], who established them on the basis of experimental measurements of distances between vocal

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 4.27: Features used in ALINE and their salience settings.

organs during speech production. 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 4.27 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 practically all other alignment algorithms assign equal weight to all features, which, although superficially more elegant, does not address the problem of unequal relevance of features.

Unlike the strictly binary feature systems, 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 by using machine-learning techniques.

Feature name	Phonological term	Numerical value
Place	[bilabial]	1.0
	[labiodental]	0.95
	[dental]	0.9
	[alveolar]	0.85
	[retroflex]	0.8
	[palato-alveolar]	0.75
	[palatal]	0.7
	[velar]	0.6
	[uvular]	0.5
	[pharyngeal]	0.3
	[glottal]	0.1
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

Table 4.28: Multivalued features and their values.

	a	i	y	n	p	r	s
a	15	8	2	−50	−56	−28	−40
i	8	15	10	−26	−32	−4	−16
y	2	10	15	−21	−27	1	−11
n	−50	−26	−21	35	9	−7	5
p	−56	−32	−27	9	35	−13	19
r	−28	−4	1	−7	−13	35	3
s	−40	−16	−11	5	19	3	35

Table 4.29: A partial similarity matrix based on multivalued features with diversified salience values.

4.7.1 Data input

Representing phonetically transcribed words using only the ASCII characters is not a trivial problem. The International Phonetic Alphabet, which is a standard for representing phonetic data, contains over 80 symbols, most of which can be modified by various diacritics. Systems that assign a numerical value to every combination, such as ISO 10646/Unicode, produce totally opaque data, which is difficult to enter and maintain.

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

In order to encode the phonetic data for ALINE, I designed a scheme in which every phonetic symbol is represented by a single lowercase letter followed by zero or more uppercase letters. The initial lowercase letter is the base letter most similar to the sound represented by the phonetic symbol. Tables 4.30 and 4.31 show the default feature assignments for the base letters. The remaining uppercase letters stand for the features in which the represented sound differs from the sound defined by the base letter. For example, the French front nasal vowel [œ̃], which occurs in the word *un*, is represented by 'oFN', where F stands for *front*, and N stands for *nasal*. The full set of codes is given in Table 4.32. The scheme has a number of advantages: it is flexible — new codes can be easily introduced; it can represent any sound that can be expressed using phonetic features; and it leads to readable and concise encoding of the data (the ratio of the number of characters used in the encoding to the number of encoded phonetic symbols is usually below 1.2). The only drawback of the scheme is its dependence on the particular feature system that is used in ALINE.

SAMPA (Speech Assessment Methods Phonetic Alphabet) has been proposed as a standard for representing phonetic segments using ASCII characters. The system is similar to the scheme used in ALINE, but, in my opinion, not nearly as practical. First, it does not include all IPA symbols. Second, the use of non-alphabetic ASCII characters makes it somewhat opaque. Finally, unlike my feature-based scheme, it requires a large lookup table that covers all phonetic symbols. It would not be difficult, however, to write a script for converting SAMPA-encoded data to the form required by ALINE.

	Place	Manner	Syl	Vce	Nas	Ret	Lat
a	velar	low vowel	1	1	0	0	0
b	bilabial	stop	0	1	0	0	0
c	alveolar	affricate	0	0	0	0	0
d	alveolar	stop	0	1	0	0	0
e	palatal	mid vowel	1	1	0	0	0
f	labiodental	fricative	0	0	0	0	0
g	velar	stop	0	1	0	0	0
h	glottal	fricative	0	0	0	0	0
i	palatal	high vowel	1	1	0	0	0
j	alveolar	affricate	0	1	0	0	0
k	velar	stop	0	0	0	0	0
l	alveolar	approximant	0	1	0	0	1
m	bilabial	stop	0	1	1	0	0
n	alveolar	stop	0	1	1	0	0
o	velar	mid vowel	1	1	0	0	0
p	bilabial	stop	0	0	0	0	0
q	glottal	stop	0	0	0	0	0
r	retroflex	approximant	0	1	0	1	0
s	alveolar	fricative	0	0	0	0	0
t	alveolar	stop	0	0	0	0	0
u	velar	high vowel	1	1	0	0	0
v	labiodental	fricative	0	1	0	0	0
w	velar (bilabial)	high vowel	1	1	0	0	0
x	velar	fricative	0	0	0	0	0
y	velar	high vowel	1	1	0	0	0
z	alveolar	fricative	0	1	0	0	0

Table 4.30: The default feature assignments for base letters.

	Place	Manner	High	Back	Round
a	velar	low vowel	low	central	0
e	palatal	mid vowel	mid	front	0
i	palatal	high vowel	high	front	0
o	velar	mid vowel	mid	back	1
u	velar	high vowel	high	back	1
w	velar (bilabial)	high vowel	high	back	1
y	velar	high vowel	high	front	0

Table 4.31: Additional feature assignments for non-syllabic segments.

Code	Feature name	Feature value
A	Aspirated	[aspirated]
B	Back	[back]
C	Back	[central]
D	Place	[dental]
F	Back	[front]
H	Long	[long]
N	Nasal	[nasal]
P	Place	[palatal]
R	Round	[round]
S	Manner	[fricative]
V	Place	[palato-alveolar]

Table 4.32: ALINE's input codes.

4.8 Evaluation

A proper evaluation of an alignment algorithm requires a "gold standard" — a sufficiently large set of alignments of nontrivial word pairs. Unfortunately, such a standard is not easily available, and constructing it from scratch would be costly and difficult. Although linguistic textbooks contain examples of genetically related words, their alignment is almost never given explicitly. The determination of the correct alignment of two remote cognates is a task that usually calls for an expert historical linguist. In addition, the compilation of test data by the author of the tested algorithm runs the risk of introducing a subconscious bias reflecting his knowledge of its strengths and weaknesses.

Instead of building my own test set, I decided to use the 82 cognate pairs compiled by Covington [1996]. The complete set, which contains mainly words from English, German, French, Spanish, and Latin, is included in Appendix D. Unfortunately, the fact that the set was also used for tuning the parameters of ALINE reduces the value of the evaluation described in this section.

Covington's data set is not a particularly good candidate for a benchmark. In addition to the errors in the phonetic transcriptions and the inconsistent selection of morphological forms, the set contains too many undemanding word pairs: how bad would an algorithm have to be to misalign *long:lang*? Nevertheless, it did become 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.

4.8.1 Qualitative evaluation

The evaluation involves the alignment algorithms of Covington [1996], Somers [1999], and Oakes [2000], as well as ALINE. 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. Appendix D contains a complete set of alignments of the 82 cognate pairs generated by ALINE side by side with Covington's alignments. In order to perform a fair and consistent comparison, I refrained from making any corrections in the set of cognates.

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 prefers to match [o] with [ə] than 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 in between. This lack of discernment sometimes causes a proliferation of alternative alignments. In the case of English *fish* and Latin *piscis* there are four of them, and without the bias against isolated indels there would be six.

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. For example, in the English/Greek pair *daughter:thugatēr*, it correctly opts for linking [d] with [t^h] rather than with [g]. 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. For example, in the Spanish/French pair *cabeza:cap*, it blindly matches a labial stop with a dental fricative rather than with another labial stop.

In spite of its comprehensive set of edit operations, Oakes's JAKARTA aligner performs poorly in comparison with the other algorithms. It does display occasional brilliance; for example, it posits a complex correspondence between [t] with [ts] in *tongue*:*Zunge*. However, it also makes elementary mistakes: it frequently aligns consonants with vowels, postulates unusual sound changes with no foundation, and has a tendency to align shorter words with the suffixes of the longer words.

With the exception of a few mistakes, ALINE does a good job both on closely and remotely related language pairs. On the Spanish/French set, ALINE makes no mistakes at all. Unlike Covington's aligner, it properly aligns [l] in *arbol* with the second [r] in *arbre*. On the English/German data, it selects the correct alignment in those cases where Covington's aligner produces two alternatives. In the final, mixed set, ALINE makes a single mistake in *daughter:thugatēr*, in which it posits a dropped prefix rather than a syncopated syllable; in all other cases, it is right on target.

The examples in Table 4.33 are taken from the English–Latin set, which is particularly interesting due to the ample diachronic distance between the two languages. ALINE correctly discards inflectional affixes in *piscis* and *flāre*, and posits the operation of compression/expansion to account for the cases of diphthongization of vowels in *I* and *tooth*. In *fish:piscis*, Covington's aligner produces four alternative alignments, while ALINE selects the correct one. Both algorithms are technically wrong

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

Table 4.33: Examples of alignments of English and Latin cognates.

on *tooth:dentis*, but this is hardly an error considering that only the information contained in the phonetic string is available to the aligners. In order to produce a historically correct alignment shown in Table 4.34, an aligner would have to know something about regular sound correspondences between English and Latin. I will return to this example in Chapter 6.

‖	t	uw	-	θ	‖	
‖	d	e	n	t	‖	is

Table 4.34: The correct alignment of *tooth:dentis*.

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

Table 4.35: Evaluation of alignment algorithms on Covington's data set.

4.8.2 Quantitative evaluation

The tables containing the generated alignments take up a substantial part of Covington's article. However, the true alignments are never explicitly given; the author only refers to them indirectly when he discusses the errors made by his program. In order to make the comparison of various algorithms more rigorous, I recreated the set of true alignments to the best of my knowledge, verifying the difficult cases in the historical linguistic literature [Buck, 1949; Hall, 1976; Bloomfield, 1946], which takes into account the results of research conducted over a period of many decades, and often also historical records that document the evolution of languages.

For the comparison, I adopted a straightforward evaluation 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. So, for example, Covington's aligner received 0.33 for the pair *ager:ajras* because one of the three proposed alignments is right. 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 4.35.

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." Oakes's JAKARTA scores well below the rest.

4.9 Conclusion

The results on a limited set of cognate pairs show that, overall, ALINE produces better alignments than Covington's and Somers's algorithms. However, the method of evaluation used in this chapter is not completely satisfactory. 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. Identification of cognates, which is the topic of Chapter 5, is an example of such a task. An alignment score of two words can be taken as a measure of their phonetic similarity, which in turn is a good indicator of the likelihood of cognation.

The alignment of corresponding segments in potential cognates is an essential step of the comparative method of language reconstruction. ALINE, as well as other alignment algorithms described in this section, aligns one word pair at a time, and has no learning ability. The lack of knowledge about regular sound correspondences limits the accuracy of the alignment. In Chapter 6, I will show how the alignment of cognates can be driven by automatically determined correspondences.

Chapter 5

Identification of cognates

In the narrow sense used in historical linguistics, cognates are words in related languages that have developed independently from the same proto-form (so-called **genetic cognates**). Due to their common origin, genetic cognates often sound alike and have the same or similar meaning. However, phonetic similarity of words in different languages can also be due to other reasons, which are illustrated in Table 5.1. In general, deciding whether two words are genetically related requires expert knowledge of the history of the languages in question.

Spanish	English	Classification
sal	*salt*	genetic cognates
suéter	*sweater*	direct borrowing
ambición	*ambition*	borrowing from a third language
mucho	*much*	chance similarity
carpeta 'folder'	*carpet*	"false friends"
cuclillo	*cuckoo*	onomatopoeic words
mamá	*mommy*	nursery words

Table 5.1: Examples of similar words in Spanish and English.

The identification of cognates is a component of two principal tasks of historical linguistics: establishing the relatedness of languages and reconstructing the histories of language families. In the past, languages were often grouped in families on the basis of similarity of basic vocabulary. Nowadays, most linguists insist on corroborating the claims of relatedness with a list of regular sound correspondences. Nevertheless, a recently developed approach of *mass comparison* [Greenberg, 1987] rejects correspondences in favour of similarity, and has produced several proposals of very remote genetic relationship.

In computational linguistics, cognates have been employed for a number of bitext-related tasks.

- sentence and word alignment in bitexts [Simard *et al.*, 1992; Church, 1993; McEnery and Oakes, 1996; Melamed, 1999],
- improving statistical machine translation models [Al-Onaizan *et al.*, 1999],
- inducing translation lexicons [Koehn and Knight, 2001; Mann and Yarowsky, 2001],
- automatic construction of weighted string similarity measures [Tiedemann, 1999],
- extracting lexicographically interesting word-pairs from multilingual corpora [Brew and McKelvie, 1996].

Most of the above approaches take advantage of the property that almost all co-occurring cognates in bitexts are mutual translations. Note that in computational linguistics, the term *cognate* usually denotes words in different languages that are similar in form and meaning, without making a distinction between borrowed and genetically related words. For example, English *sprint* and the Japanese loan *supur-*

into are considered cognate for this purpose, even though these two languages are unrelated. This broad definition is also adopted in this chapter.

The task addressed in this chapter can be formulated in two ways. On the word level, given two words (*lexemes*) from different languages, the goal is to compute a value that reflects the likelihood of the pair being cognate. I assume that each lexeme is given in a phonetic notation, and that it is accompanied by one or more glosses that specify its meaning in a metalanguage for which a lexical resource is available (for example, English). On the language level, given two vocabulary lists representing two languages, the goal is to order all pairs of vocabulary entries according to their likelihood of cognation. Tables 5.2 and 5.3 show sample entries from two typical vocabulary lists. Such vocabulary lists are sometimes the only data available for lesser-studied languages.

In the traditional approach to cognate identification, words with similar meanings are placed side by side. Those pairs that exhibit some phonological similarity are analyzed in order to find systematic correspondences of sounds. The correspondences in turn can be used to distinguish between genuine cognates and borrowings or chance resemblances.

The approach adopted in this chapter is based on the intuition that, in spite of the inevitable diachronic changes, cognates on average display higher semantic and phonetic similarity than words that are unrelated. In the following sections, I discuss various ways of measuring phonetic and semantic similarity, and then present COGIT, a similarity-based cognate-identification system. COGIT combines ALINE, the program described in Chapter 4, with a novel procedure for detecting semantic relatedness from glosses that employs keyword selection and WordNet. Even though the similarity-based approach does not differentiate between cognates and borrowings, the experiments on data from four native American languages show that it can be

āniskōhōčikan	string of beads tied end to end
asikan	sock, stocking
kamāmakos	butterfly
kostāčīwin	terror, fear
misiyēw	large partridge, hen, fowl
namēhpin	wild ginger
napakihtak	board
tēhtēw	green toad
wayakēskw	bark

Table 5.2: An excerpt from a Cree vocabulary list [Hewson, 1999].

āšikan	dock, bridge
anaka'ēkkw	bark
kipaskosikan	medicine to induce clotting
kottāčīwin	fear, alarm
mēmīkwan'	butterfly
misissē	turkey
namēpin	sucker
napakissakw	plank
tēntē	very big toad

Table 5.3: An excerpt from an Ojibwa vocabulary list [Hewson, 1999].

successfully applied to the task of cognate identification.

5.1 Phonetic similarity

The approaches to measuring word similarity can be divided into two groups. The *orthographic* approaches disregard the fact that alphabetic symbols express actual sounds, employing a binary identity function on the level of character comparison. A one-to-one encoding of symbols has no effect on the results. The *phonetic* approaches, on the other hand, attempt to take advantage of the phonetic characteristics of individual sounds in order to estimate their similarity. This presupposes a transcription of the words into a phonetic or phonemic representation.

5.1.1 The orthographic approaches

Simard *et al.* [1992] consider two words to be cognate if they are at least four characters long and their first four characters are identical. Naturally, such a crude approach will fail to identify even very similar cognates, such as *government* and *gouvernement*. On the other hand, the simple condition can be checked very quickly. This approach can be generalized to yield a *truncation coefficient* in the $[0, 1]$ range. One way of doing this is to divide the length of the longest common prefix by the average of the lengths of the two words being compared. For example, the truncation coefficient of English *colour* and French *couleur* is $\frac{2}{6.5} \simeq 0.31$ because their longest common prefix is "co-". The truncation coefficient can be computed in $O(n)$ time.

Dice's similarity coefficient, originally developed for the comparison of biological specimens, was first used to compare words by Adamson and Boreham [1974]. It is based on the notion of a *bigram* — an ordered pair of characters. Adamson and Boreham chose bigrams rather than single characters expressly to preserve information about the sequence of the letters in a word. Dice's coefficient was adopted for cognate identification by McEnery and Oakes [1996], who concluded that it performed better on the task than a measure based on a simple edit distance. Brew and McKelvie [1996] proposed four other bigram-based methods. Dice's coefficient is determined by the ratio of the number of shared character bigrams to the total number of bigrams in both words:

$$DICE(x,y) = \frac{2|bigrams(x) \cap bigrams(y)|}{|bigrams(x)| + |bigrams(y)|},$$

where *bigrams(x)* is a multi-set of character bigrams in x. For example, *colour* and *couleur* share three bigrams (*co*, *ou*, and *ur*), so their Dice's coefficient is $\frac{6}{11} \simeq 0.55$. Notice that the number of bigrams in a word is one less than its length measured in characters. Dice's coefficient can be computed in $O(n^2)$ time.

Melamed [1999] detects orthographic cognates by thresholding the Longest Common Subsequence Ratio (LCSR). The LCSR of two words is computed by dividing the length of their longest common subsequence (LCS) by the length of the longer word:

$$LCSR(x,y) = \frac{|LCS(x,y)|}{max(|x|,|y|)}.$$

For example, LCSR(*colour*,*couleur*) $= \frac{5}{7} \simeq 0.71$, as their longest common subsequence is "c-o-l-u-r". Brew and McKelvie [1996] propose a variation in which the denominator is the average of both word lengths. Melamed explicitly chooses LCSR over DICE and truncation, but he does not provide any experimental results to support his preference. A simple dynamic programming algorithm computes the LCSR in $O(n^2)$ time, and more complicated algorithms are even faster.

It is worth noting that LCSR is closely related to edit distance. If the cost of a substitution is set at more than twice the cost of an indel, the calculation of an optimal alignment of the two strings is equivalent to finding their longest common subsequence [Kruskal, 1983, page 30], and the edit distance d can be expressed in terms of LCS:

$$d(x, y) = |x| + |y| - |2 \times LCS(x, y)|.$$

5.1.2 The phonetic approaches

The approaches of Kessler [1995] and Nerbonne and Heeringa [1997] to estimating phonetic distance between dialects have already been discussed in Chapter 4. Kessler's surprising finding was that a binary distance function based on the identity of phonetic segments worked better than a feature-based measure. Nerbonne and Heeringa's conclusion was the opposite. It has to be noted that in neither case was the evaluation method totally objective: Kessler calculated the correlation with a distance measure based on the number of isoglosses in a linguistic atlas, while Nerbonne and Heeringa apparently visually compared their results with a dialect map. In any case, the task of computing the distance between wordlists, albeit related, is clearly distinct from the task of identifying cognates *in* wordlists.

Oakes [2000], on the other hand, focuses directly on cognate identification. His approach is very simple: two words are deemed to be cognate if their edit distance is below a certain threshold; the length of the words is irrelevant. The threshold was established by the analysis of the distances between cognate and non-cognate pairs in four Indonesian wordlists.

ALINE (described in Chapter 4) was originally developed for aligning phonetic strings, but since it chooses the optimal alignment on the basis of a similarity score, no substantial modification is necessary for the current application. The similarity score returned by ALINE can be normalized by dividing it by the length of the longer word multiplied by the maximum possible similarity score between segments, so that it falls in the range $[0,1]$. Intuitively, a complex algorithm such as ALINE should be more accurate than simple, orthographic coefficients. By applying various methods to a specific task, such as cognate identification, their relative performance can be objectively evaluated.

5.2 Semantic similarity

Since the meanings of the lexemes are given by their glosses, the simplest method to detect semantic similarity is to check if the lexemes have at least one gloss in common. For example, the cognates *kottāčīwin* 'terror, fear' and *kostāčīwin* 'fear, alarm' in Tables 5.2 and 5.3 are correctly associated by this method. However, in many cases, the similarity of semantically related glosses is not recognized by this method. The most common reasons are listed below with examples.

1. Spelling errors or variants: 'vermilion' and 'vermillion', 'sweet grass' and 'sweetgrass', 'plow' and 'plough';
2. Morphological differences: 'ash' and 'ashes';
3. Determiners: 'a mark' and 'mark', 'my finger' and 'finger', 'fish' and 'kind of fish';
4. Adjectival modifiers: 'small stone' and 'stone';
5. Nominal modifiers: 'goose' and 'snow goose';
6. Complements and adjuncts: 'stone' and 'stone of peach', 'island' and 'island in a river';
7. Synonymy: 'grave' and 'tomb';

8. Small semantic changes: ‘fowl’ and ‘turkey’;

9. Radical semantic changes: ‘broth’ and ‘grease’.

Spelling errors, which may be especially frequent in data that have been acquired through optical character recognition, are easy to detect but have to be corrected manually. Morphological differences (category 2) can be removed by lemmatization. Many of the cases belonging to categories 3 and 4 can be handled by adopting a stop-list of determiners, possessive pronouns, and very common modifiers such as *certain, kind of, his, big, female,* etc.

Categories 4, 5, and 6 illustrate a common phenomenon of minor semantic shifts that can be detected without resorting to a lexical resource. All that is needed is the determination of the heads of the phrases, or, more generally, *keywords.* Pairs of glosses that contain matching keywords are usually semantically related.

For the remaining categories, string matching is of no assistance, and some lexical resource is called for.

5.2.1 WordNet

WordNet [Fellbaum, 1998] is a publicly available lexical knowledge base for English developed at Princeton University. Lexical entries are organized into comprehensive networks of synonym sets (*synsets*). Words that have more than one meaning (*polysemous* words) may participate in several different synsets. The synsets are linked by various *lexical relations.* The noun database is the most developed part of the semantic network. The main types of lexical links between noun synsets are shown in Table 5.4. The hypernymy/hyponymy links form the *hierarchy* of nouns. They link each synset to its immediately more general and more specific synsets. A chain of

Type	Name	Example	Inverse of
hypernymy	IS-A	*bird* $\rightarrow$ *animal*	hyponymy
hyponymy	SUBSUMES	*bird* $\rightarrow$ *robin*	hypernymy
meronymy	PART-OF	*beak* $\rightarrow$ *bird*	holonymy
holonymy	HAS-A	*tree* $\rightarrow$ *branch*	meronymy
antonymy	COMPLEMENT-OF	*leader* $\leftrightarrow$ *follower*	*itself*

Table 5.4: The main lexical relations between nouns in WordNet.

hypernymy links can be traversed from each synset to one of the 11 abstract concepts that are at the top of the hierarchy.

The coverage of WordNet is impressive. In version 1.6, the noun network alone is made of over $60,000$ synsets, which contain over $100,000$ noun senses, and the number of lexical links exceeds $150,000$. Nonetheless, the coverage of the English vocabulary is inevitably only partial.

The idea of using WordNet for the detection of semantic relationships comes from Lowe and Mazaudon [1994, footnote 13, page 406]. WordNet is well-suited not only for detecting synonyms but also for associating lexemes that have undergone small semantic changes. Trask [1996] lists several types of semantic change, including the following:

- **generalization** (broadening): 'partridge' $\rightarrow$ 'bird';
- **specialization** (narrowing): 'berry' $\rightarrow$ 'raspberry';
- **melioration** (developing a more favourable sense): 'woman' $\rightarrow$ 'queen';
- **pejoration** (developing a less favourable sense): 'farm-worker' $\rightarrow$ 'villain';
- **metaphor** (extending the literal meaning): 'steersman' $\rightarrow$ 'governor';

- **metonymy** (using an attribute of an entity to denote the entity itself): 'crown' → 'king';
- **synechdoche** (using a part to denote a whole, or vice-versa): 'hand' → 'sailor'.

Certain types of semantic change have direct parallels among WordNet's lexical relations. *Generalization* can be seen as moving up the IS-A hierarchy along a hypernymy link, while *specialization* is moving in the opposite direction, along a hyponymy link. *Synecdoche* can be interpreted as a movement along a meronymy/holonymy link. However, other types of semantic change, such as metonymy, melioration/pejoration, and metaphor, have no direct analogues in WordNet.

The use of WordNet for semantic similarity detection is possible only if English is the glossing metalanguage. If the available vocabularies are glossed in other languages, one possible solution is to translate the glosses into English, which, however, may increase their ambiguity. A better solution could be to use a multilingual lexical resource, such as EuroWordNet [Vossen, 1998], which is modeled on the original Princeton WordNet. In general, WordNet could be replaced by any machine-readable dictionary or thesaurus, or even word-frequency data extracted from a corpus.

5.3 Implementation

COGIT, the cognate identification system, is schematically depicted in Figure 5.1. The system is composed of a set of *Perl* scripts for preprocessing the vocabulary lists, and phonetic and semantic modules written in C++. Given two vocabulary lists representing distinct languages, COGIT produces a list of vocabulary-entry pairs, sorted according to the estimated likelihood of cognation. In order to take full advantage of WordNet's noun hierarchies, the vocabulary lists are restricted to contain only nouns.

1. For each gloss in vocabularies L_1 and L_2:
   (a) Remove stop words.
   (b) Select keywords.
   (c) Perform lemmatization.
   (d) Generate lists of semantically related words.
2. For each pair of entries $((l_i, g_i), (l_j, g_j)) \in (L_1 \times L_2)$:
   (a) Compute the phonetic similarity score $Sim_{phon}(l_i, l_j)$.
   (b) Compute the semantic similarity score $Sim_{sem}(g_i, g_j)$.
   (c) $Sim_{overall}(i, j) \leftarrow (1 - \alpha)\cdot\ Sim_{phon}(l_i, l_j) + \alpha\cdot\ Sim_{sem}(g_i, g_j)$.
   (d) If $Sim_{overall}(i, j) \geq T$, record i, j, and $Sim_{overall}(i, j)$.
3. Sort the pairs in descending order of $Sim_{overall}$.

Table 5.5: Cognate identification algorithm.

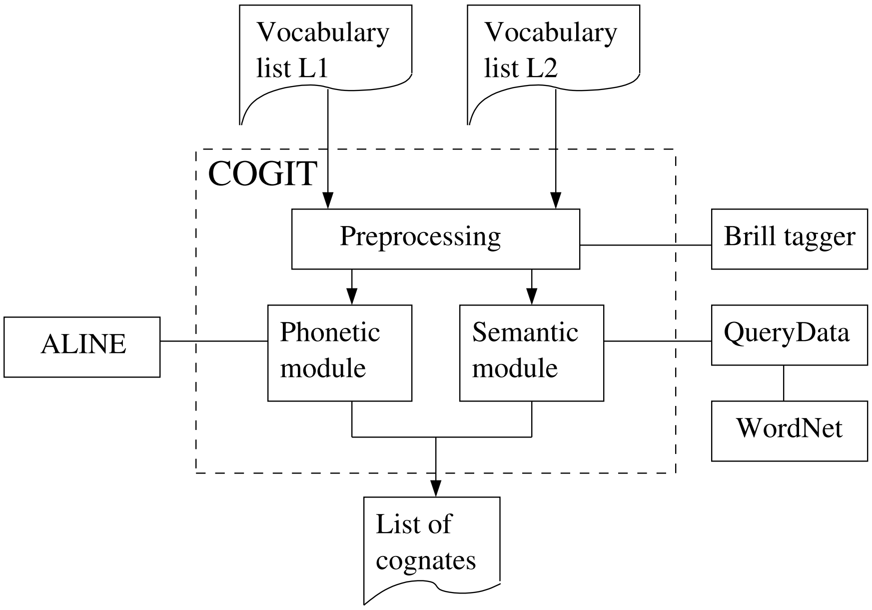

Figure 5.1: The structure of cognate identification system.

The algorithm is presented informally in Table 5.5. Each vocabulary entry consists of a lexeme l and its gloss g. The overall similarity score is a linear combination of the similarity scores returned by the phonetic and semantic modules. The parameter α reflects the relative importance of the semantic vs. phonetic score.

The preprocessing of the vocabulary data is performed by a suite of *Perl* scripts. Checking and correcting the spelling of glosses is assumed to have been done beforehand. Table 5.6 shows the complete list of stop words and phrases that are removed from glosses in order to facilitate the matching of nearly-identical expressions. Glosses that exceed 30 characters are truncated, and all comments enclosed in brackets are deleted. The lemmatization process is carried out by the QueryData module, which is described below. The phonetic similarity score can be computed either by ALINE or through one of the orthographic methods described in section 5.1.1.

The keyword selection is performed according to a simple heuristic with the aid of a part-of-speech tagger [Brill, 1995]. Since the tagger operates on sentences rather than on phrases, all glosses are initially prepended with the string 'It is a'. The string is removed after the tagging process is completed. Since the lexemes are restricted to nouns, only words with the NN tag are considered as possible keywords, except when a gloss contains a single word, in which case the word is taken to be the keyword regardless of the tag. The gloss is scanned from left to right, and all nouns are marked as keywords unless a wh-word or a preposition other than 'of' is encountered.

Table 5.7 contains examples of keyword selection in action. Keywords are under-

a	an	the	my	your
his	her	very	large	small
big	little	male	female	certain
kind of	manner of	piece of		

Table 5.6: The stop-list of words that are removed from glosses.

‘ $\underline{string^{NN}}$ for^{IN} $stretching^{VBG}$ $hide^{NN}$ ’

‘ $upright^{JJ}$ $\underline{ornament^{NN}}$ $worn^{VBN}$ on^{IN} $head^{NN}$ ’

‘ $yellow^{JJ}$ $\underline{feather^{NN}}$ $with^{IN}$ $black^{JJ}$ tip^{NN} ’

‘ $\underline{sorcerer^{NN}}$ who^{WP} has^{VBZ} a $serpent^{NN}$ ’

‘ $\underline{clot^{NN}}$ of^{IN} $\underline{blood^{NN}}$ ’

‘ $\underline{flint^{NN}}$, $detonating^{VBG}$ $\underline{cap^{NN}}$ on^{IN} $cartridge^{NN}$ ’

‘ $\underline{snow^{NN}}$ $\underline{dart^{NN}}$, $\underline{ice^{NN}}$ $throwing^{VBG}$ $stick^{VB}$ ’

‘ $\underline{sign^{NN}}$ $which^{WDT}$ $points^{NNS}$ the way^{NN} ’

‘ a $\underline{portage^{NN}}$, $setting^{VBG}$ $ashore^{RB}$ ’

‘ little $\underline{story^{NN}}$ $that^{WDT}$ is^{VBZ} $sometimes^{RB}$ $told^{VBN}$ ’

‘ $\underline{mysterious^{JJ}}$, $haunted^{VBN}$ $\underline{person^{NN}}$ or^{CC} $\underline{place^{NN}}$ ’

Table 5.7: Examples of automatically tagged glosses with keywords marked.

lined. Words are accompanied by the assigned part-of-speech tags: prepositions are tagged as ‘IN’, while wh-words have tags that start with ‘W’. Stop-words are shown in a distinct font. It is evident from the handful of examples that the keyword scheme is far from perfect. Because of the limited accuracy of the part-of-speech tagger, some words are mistagged to begin with (e.g. ‘stick’). A comma separating two adjectives, as in ‘mysterious, haunted’ is indistinguishable from a comma separating two alternative glosses, so ‘mysterious’ is erroneously assumed to be an independent gloss. Nevertheless, the heuristic seems to pick most of the relevant nouns with reasonable precision.

For the calculation of a WordNet-based semantic similarity score, I initially used the length of the shortest path between synsets, measured in the number of IS-A links. One can imagine utilizing a more sophisticated measure of semantic similarity, such as

Leacock and Chodorow's [1998] normalized path length, Resnik's [1999] information-based approach, Lin's [1998] universal similarity measure, or other methods described in Budanitsky [1999]. However, I found the effect of considering paths longer than one link to be negligible. Moreover, the process of determining the link distances between all possible pairs of glosses, separately for each pair, was too time-consuming.

The solution eventually adopted is to compute the semantic score by a faster method that employs QueryData, a *Perl* interface to the WordNet module [Rennie, 1999]. A list of synonyms, hypernyms, and meronyms is generated for each gloss and keyword in the preprocessing phase. During the execution of the core C++ program, regular string matching is performed directly on the listed senses. Words are considered to be related if there is a relationship link between any of their senses. The semantic score is determined according to a 9-point scale of semantic similarity, which is shown in Table 5.8. The levels of similarity are considered in order, starting with gloss identity. The scores are not cumulative. For example, if the program detects that a keyword in one gloss is a hypernym of a keyword in another gloss ("keyword hypernymy"), the check for a meronymy relation between the entire glosses ("gloss meronymy") is not performed. The exact values corresponding to each level were established by trial and error using the development set.

Let us follow the entire process of computing the similarity between vocabulary entries on an example involving Cree *wāhkwa* 'a lump of roe' and Ojibwa *wākk* 'fish eggs'. After the preprocessing removes the determiner a from the first gloss, the glosses are tagged and four nouns are identified as keywords in the respective glosses: lump, roe, fish, eggs. The lemmatization removes the plural ending -s from eggs. Neither of the complete glosses exists in WordNet, but each of the keywords is represented by several senses. The WordNet sense lists for the keywords are shown in Table 5.9. COGIT detects two semantic relations: roe is a kind of egg, and roe is a part of

Rank	Similarity level	Score
1	gloss identity	1.00
2	gloss synonymy	0.70
3	keyword identity	0.50
4	gloss hypernymy	0.50
5	keyword synonymy	0.35
6	keyword hypernymy	0.25
7	gloss meronymy	0.10
8	keyword meronymy	0.05
9	none detected	0.00

Table 5.8: Semantic similarity levels.

fish. Since keyword hypernymy has a higher precedence than keyword meronymy, the resulting semantic score is 0.25. The combination with the phonetic score of 0.4167 between *wāhkwa* and *wākk* with α set to 0.2 produces an overall similarity coefficient of 0.3834.

5.4 Evaluation

The test data suitable for the evaluation of the approach outlined above has to fulfill several requirements. It should be sufficiently large to contain most of the surviving cognates. The lexemes should be given in a consistent notation that allows for an automatic transcription into phonetic form. In order to take advantage of WordNet, the glosses must be given in English. Finally, the cognation information has to be provided in the electronic form as well, so that the performance of the program can be measured objectively. The last condition is perhaps the most difficult to satisfy. Even in the rare cases when machine-readable bilingual lexicons can be acquired, the cognation judgements would have to be laboriously extracted from etymological

Word	Synonyms	Hyponyms	Meronyms
lump	ball, clod, glob, clump, chunk, swelling, klutz, puffiness, lout, clod, goon, stumblebum, oaf, lubber, lummox, gawk, hunk	agglomeration, piece, part, symptom, clumsy person	
roe	hard roe	spawn, **egg**, seafood	**fish**
fish	chump, fool, gull, mark, patsy, fall guy, sucker, shlemiel, soft touch, mug, go fish	foodstuff, food product, victim, card game, cards, dupe, aquatic vertebrate	pisces, school, shoal
egg	testis, gonad, testicle, ball, ballock, bollock, nut	endocrine gland, ductless gland, ovum, egg cell, foodstuff, food product	male genitalia, family jewels

Table 5.9: Lists of semantically related words generated from WordNet.

dictionaries. Note that optical scanning of phonetic symbols or unusual diacritics is practically impossible.

In that situation, it was truly munificent of John Hewson to provide me with his Algonquian data.

5.4.1 The Algonquian data

The machine-readable Algonquian data consists of two parts that complement each other: the etymological dictionary, and the vocabulary lists from which the dictionary was produced. The dictionary is available in book form [Hewson, 1993]. The computer files that I obtained from John Hewson contain both the dictionary and the raw lists.

The dictionary contains 4,068 cognate sets, including 853 marked as nouns. Two typical cognate sets are shown in Table 5.10. Each set is composed of a reconstructed proto-form and the corresponding cognates. The first line contains the reconstructed lexeme, a two letter code indicating its grammatical category, the English gloss, the

```
*kenwe:cye:wa                  NA pike                             1027
    C    kinose:w                       fish
    F    keno:ce:wa                     pike
    M    kenu:si:w                      pickerel
    O    kino:ce:                       pike (kind of fish)
*kenwehkwetamwa                TI cut long    (*-ehkwet)           1028
    C    kinohkotam                     he cuts it long
    O    kino:kkota:n                   cut long(er)
```

Table 5.10: An excerpt from the Algonquian dictionary.

major morpheme of the lexeme (optional), and the lexeme number. The subsequent lines describe the surviving cognates. Each line includes a one-letter language code, the cognate lexeme, and its English gloss. Nearly all cognates belong to one of the four principal Algonquian languages (Fox, Menomini, Cree, Ojibwa).

The dictionary file is almost identical with the book version, and required only minimal clean-up. The lexemes are already in a phonemic transcription, so no sophisticated grapheme-to-phoneme conversion was necessary. A simple coding is used to express phonemes that lack ASCII equivalents: *c* for [š], *3* for [ɛ], etc. In the experiments described in this section, the dictionary file served as a source of the cognation information ("gold standard").

In contrast with the dictionary, the vocabulary lists can be characterized as noisy data. They contain many errors, inconsistencies, duplicates, and lacunae. The Fox file is incomplete. In the Menomini file, three different phonemes ([č], [ɛ], and [ʔ]) had been merged into one, and had to be painstakingly reconstructed on the basis of phonotactic constraints. As much as possible, the entries were cross-checked with the dictionary itself, which is much more consistent. Table 5.11 specifies the number of unique lexemes available for each language. It appears that only about a third of the nouns present in the vocabulary lists had made it into the dictionary.

Language	Dictionary only		Dictionary and lists	
	All words	Nouns	All words	Nouns
Fox	1252	193	4759	575
Menomini	2231	361	8550	1540
Cree	2541	512	7507	1628
Ojibwa	2758	535	6334	1023
Total	**8782**	**1601**	**27150**	**4766**

Table 5.11: The size of the vocabulary lists.

5.4.2 Properties of the data

Table 5.12 shows, for each language combination, the number of cognate noun pairs extracted from the dictionary, and the total number of possible pairings of lexemes, which was calculated by multiplying the number of entries in the respective vocabularies. To take the Menomini–Ojibwa pair as an example, the task of the system was to identify 259 cognate-pairs from 1540×1023 possible lexeme-pairs. On average, the ratio of non-cognate to cognate pairs was about 6500, which gives an idea of the difficulty of the task of identifying cognate pairs.

The values in Tables 5.13 and 5.14 confirm that both the phonetic and the semantic similarity between Algonquian cognates (C) is greater than between randomly selected lexemes (R). The average weighted phonetic similarity between cognates can be seen as a measure of the relative “closeness” of a pair of languages. According to all measures, Cree–Ojibwa is the most closely related pair, Fox is somewhat removed from the two, and Menomini is even more distant.

Figure 5.2 compares proportions of all cognate pairs in the data that are covered by individual semantic similarity levels. Over 60% of cognates have at least one

Languages		Cognates	Word pairs
Fox	Menomini	121	885,500
Fox	Cree	130	936,100
Fox	Ojibwa	136	588,225
Menomini	Cree	239	2,507,120
Menomini	Ojibwa	259	1,575,420
Cree	Ojibwa	408	1,665,444

Table 5.12: The number of shared cognates and the number of possible word pairs for each language combination (nouns only).

Languages		Trunc.		DICE		LCSR		ALINE	
		R	C	R	C	R	C	R	C
Fox	Menomini	.013	.247	.058	.343	.229	.570	.220	.607
Fox	Cree	.012	.290	.067	.466	.248	.633	.223	.616
Fox	Ojibwa	.013	.323	.060	.501	.236	.651	.212	.626
Menomini	Cree	.012	.266	.055	.316	.227	.599	.219	.620
Menomini	Ojibwa	.012	.216	.046	.277	.212	.551	.206	.590
Cree	Ojibwa	.013	.359	.083	.618	.255	.768	.224	.699

Table 5.13: Average phonetic similarity values computed by various methods for randomly selected word pairs and for cognate pairs.

Languages		Random	Cognates
Fox	Menomini	.003	.753
Fox	Cree	.002	.719
Fox	Ojibwa	.002	.710
Menomini	Cree	.003	.698
Menomini	Ojibwa	.003	.718
Cree	Ojibwa	.002	.681

Table 5.14: Average semantic similarity values for randomly selected word pairs and for cognate pairs.

gloss in common. The cases in which the existence of a WordNet relation influences the value of the similarity score account for less than 10% of the cognate pairs. In particular, instances of meronymy between cognates are very rare.

5.4.3 Performance of COGIT

The Cree–Ojibwa language pair was chosen as the development set. These two languages are represented by the most complete vocabularies and share the largest number of cognates. However, they also turned out to be the most closely related among the four Algonquian languages, according to all measures of phonetic similarity (Table 5.13). It is quite possible that the overall performance of the system would have been better if a different language pair had been chosen as the development set.

The values of all parameters, including α, ALINE's parameters, and the semantic similarity scale given in Table 5.8, were established during the development phase of the system, using only the Cree–Ojibwa data. The optimal value of α was found to be near 0.2. Similarly, ALINE's parameters were set as follows: $C_{skip} = –1$, $C_{sub} = 10$, $C_{exp} = 15$, and $C_{vwl} = 1$. The salience settings were as given in Chapter 4, except

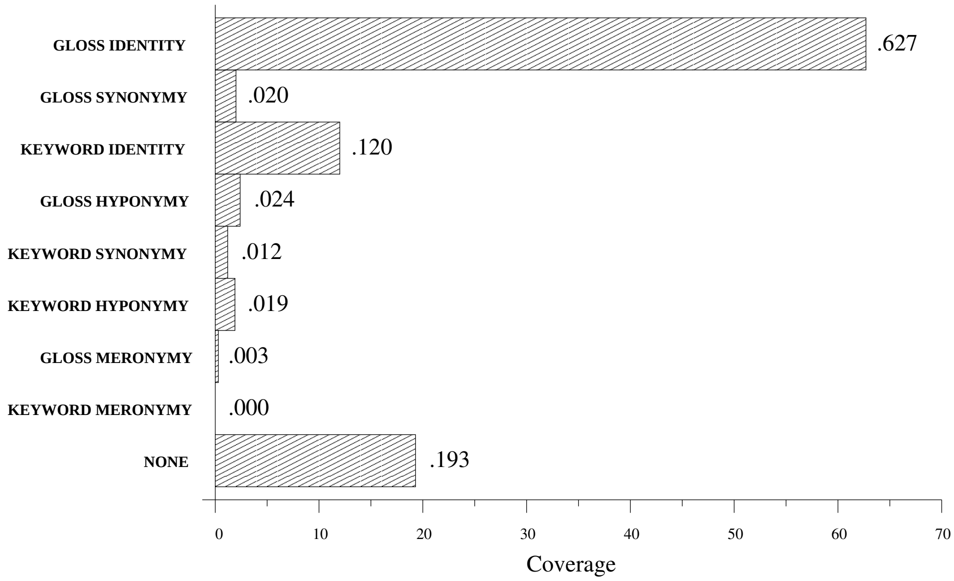

Figure 5.2: Coverage of the similarity levels.

that the salience of feature *Long* was set to 5. The semantic similarity scores were approximated by inducing a decision tree on the data and interpolating the phonetic similarity thresholds for each level.

The 11-point interpolated average precision was adopted to measure the effectiveness of the various cognate identification methods. The output of the system is a list of suspected cognate pairs sorted by their similarity scores. Typically, true cognates are very frequent near the top of the list, and become less frequent towards the bottom. The threshold value that determines the cut-off depends on the intended application, the degree of relatedness between languages, and the particular method used. Rather than reporting precision and recall values for an arbitrarily selected threshold, precision is computed for the recall levels of 0%, 10%, 20%, ..., 100%, and then averaged to yield a single number. In the experiments reported below, I uniformly assumed the precision value at 0% recall to be 1, and the precision value at 100% recall to be 0.

Languages		Trunc.	DICE	LCSR	ALINE
Fox	Menomini	.122	.113	.163	.383
Fox	Cree	.196	.215	.303	.424
Fox	Ojibwa	.235	.277	.382	.508
Menomini	Cree	.122	.129	.245	.389
Menomini	Ojibwa	.108	.121	.202	.375
Average on test set		**.157**	**.171**	**.259**	**.416**
Cree	Ojibwa	.239	.430	.592	.619

Table 5.15: Average precision for various phonetic methods.

Table 5.15 compares the average precision achieved by various methods on the six language pairs. The results on the development set (Cree–Ojibwa) are separated from the results on the test set (the remaining five pairs). ALINE outperforms all orthographic coefficients, including LCSR. The dominance of ALINE is more pronounced for more remote pairs, such as Fox–Menomini. Dice's coefficient performs poorly as a cognate identification method, being only slightly better than a naive truncation method.

Table 5.16 and 5.17 show the results when semantic information is combined with LCSR and ALINE, respectively. Methods G, K, and W represent increasingly sophisticated semantic similarity detection: Method G considers gloss identity only, Method K adds keyword-matching, and Method W employs also WordNet relations. All methods that use the semantic information provided by the glosses perform substantially better than the purely phonetic methods. Impressive results are reached even when only gloss identity is considered. Adding keyword-matching and WordNet relations brings additional, albeit modest, improvements. Again, ALINE does better

Languages		Semantic method			
		None	G	K	W
Fox	Menomini	.383	.579	.616	.630
Fox	Cree	.424	.631	.636	.655
Fox	Ojibwa	.508	.633	.655	.660
Menomini	Cree	.389	.548	.562	.569
Menomini	Ojibwa	.375	.554	.588	.598
Average on test set		**.416**	**.589**	**.611**	**.622**
Cree	Ojibwa	.619	.739	.750	.762

Table 5.16: Average precision for ALINE combined with various semantic methods.

than LCSR.

Figure 5.3 shows a more detailed comparison of the effectiveness of various methods on test sets, in the form of precision–recall curves. The figure allows one to determine an approximate precision level for a broad range of recall levels, and vice versa. For example, Method W is able to identify almost 75% of all cognate pairs if we are willing to accept a one-to-one ratio of true to false positives (50% precision). The curve for Method K, which would be slightly below the curve for Method W, is omitted for clarity. The staircase-shaped curve labeled "Semantic" illustrates the performance of a purely semantic method, in which only the similarity of glosses is considered. Each step corresponds to a semantic similarity level. For example, selecting all cases of gloss identity yields .627 recall at .237 precision.

Figure 5.4 illustrates the effect of varying the setting of the parameter α on the average precision of COGIT when Method W is used. Recall that the value of α reflects the relative importance of the semantic vs. phonetic similarity. The average precision

Languages		Semantic method			
		None	G	K	W
Fox	Menomini	.163	.460	.494	.498
Fox	Cree	.303	.569	.564	.583
Fox	Ojibwa	.382	.596	.615	.624
Menomini	Cree	.245	.479	.494	.504
Menomini	Ojibwa	.202	.414	.473	.479
Average on test set		**.259**	**.504**	**.528**	**.538**
Cree	Ojibwa	.592	.727	.743	.749

Table 5.17: Average precision for LCSR combined with various semantic methods.

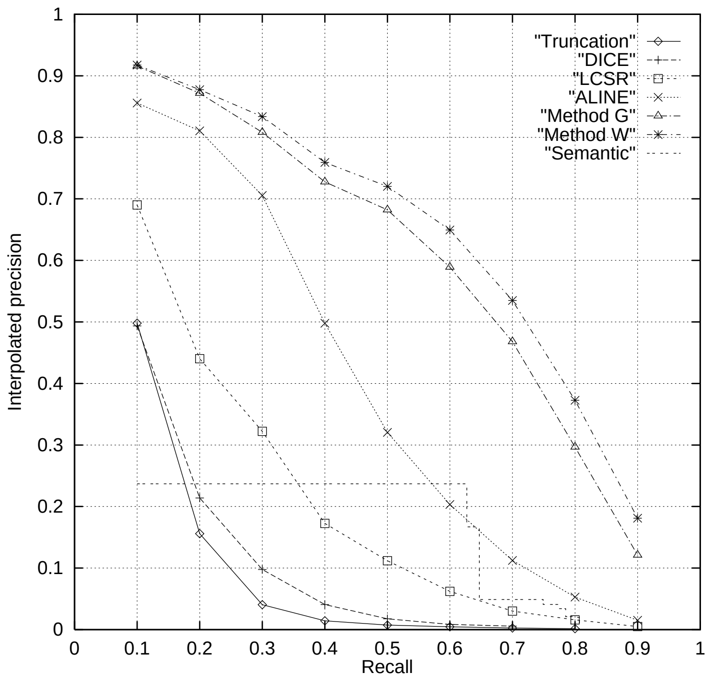

Figure 5.3: Precision-recall curves for various methods.

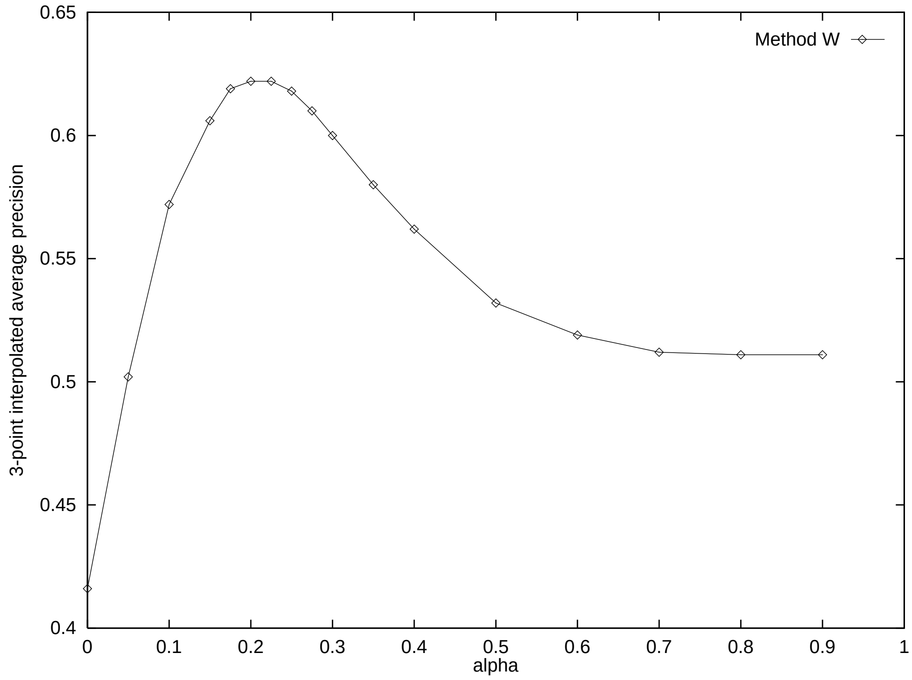

Figure 5.4: Interpolated 3-point average precision of Method W on test sets as a function of the parameter α.

for $\alpha = 0$ is .416, which represents the performance of ALINE (cf. Table 5.15). The optimal value of α for both the development and the test sets is close to 0.2. Intuitively, more weight has to be given to the phonetic score because phonetic similarity between cognates is more reliably detected than their semantic relationship. With α approaching 1, the performance levels off at .511, as precedence is increasingly given to the semantic score, which divides all pairs into nine classes corresponding to the scale of Table 5.8. When α is set to 1, Method W reduces to the pure "Semantic" approach of Figure 5.3: because the phonetic score is no longer available to order candidate pairs within semantic classes, the average precision plummets to .161.

5.5 Discussion

5.5.1 The role of WordNet

The contribution of WordNet to the overall performance of the system is relatively small. This is due both to the properties of the test data and to WordNet's shortcomings. Since the data for all Algonquian languages originates from a single project, it is quite homogeneous. As a result, many glosses match perfectly within cognate sets, which limits the need for application of WordNet lexical relations.

Another problem is that even after preprocessing that includes checking the spelling, lemmatization, and stop word removal, many of the glosses are not in a form that can be recognized by WordNet. Some rare words, such as 'spawner', 'windigo', 'cradleboard', 'smartweed', and 'spotfish', are simply not in WordNet. Compound words that are recorded in WordNet as two words, such as 'sweet grass' or 'powder horn' are not identified if written as a single word.

When many words have several meanings that participate in different synsets, the senses detected to be related are not necessarily the senses used in the glosses. For example, star and lead share a synset ('an actor who plays a principal role'), but in the Algonquian vocabularies both words are always used in their most literal sense. Only in the case of complete identity of glosses can the lexemes be assumed to be synonymous in all senses. On the other hand, words that are semantically very similar, such as puppy and dog, are often far from each other in the WordNet hierarchy.

Apart from synonymy, two other semantic relations play a role. By following the meronymy (*is-a part-of*) links, COGIT successfully associated such concepts as mattress/bed, flour/bread, and bowstring/bow. However, it also linked gun (a pedal that controls the throttle valve) with airplane, and ear in 'ear of corn' with head in 'ox, head of cattle'. Examples of concepts correctly associated by the hypernymy (*is-a-kind-of*) links include gooseberry/currant, kindness/goodness, sea/body of water, but snare/drum is off target (one of the senses of snare is 'a small drum with two heads and a snare stretched across the lower head').

In the majority of cases, the semantic associations made by WordNet are plausible: acorn *is* a part of oak, mud *is* a kind of soil, and spittle *is* the same thing as saliva. Unfortunately, most of such pairs are not cognates. The detection of semantic similarity can help, but the main burden of cognate identification has to be carried by the phonetic module.

5.5.2 False positives

The performance figures are adversely affected by the presence of the usual "noise", which is unavoidable in the case of authentic data. Manual preparation of the vocabulary lists would undoubtedly result in better performance. Nevertheless, I decided against correcting individual entries or cognation judgements. One reason was that a comprehensive clean-up of the data of such a size would not only be extremely tedious and time-consuming, but would also require advanced expertise in the field of Algonquian linguistics. Moreover, I felt that a noisy data set provides a more trustworthy test for a cognate identification program designed to help solve real linguistic problems. Only limited automatic validation of the data had been performed: the entries marked as doubtful were omitted; in the case of multiple entries of the same lexeme, a unique entry was selected, with the preference given to the entries included in the published dictionary.

It is not always easy to decide whether two similar entries are duplicates or legitimate variants. For example, Hewson's dictionary distinguishes three distinct cognate sets that correspond to the following reconstructed Proto-Algonquian etyma: **menehsyi*, **menehš-*, and **mene'tekwi*. The three proto-forms, which are uniformly glossed as 'island', have seven extant reflexes in four languages, COGIT cannot be faulted for identifying all 34 resulting cross-language combinations as probable cognates, but only five of those are counted as correct guesses because the remaining ones span different cognate sets.

In the case of **mahkahkwi* 'box', there is just one cognate set containing words from each of the four Algonquian languages, including Fox *mahkahkwi*. However, the Fox vocabulary lists contain another similar entry, *mahkahkōhi* 'little box, pail', which did not make it into the dictionary, perhaps because it is a diminutive variant of the other word. Clearly, *mahkahkōhi* is cognate with other reflexes of **mahkahkwi*. Nevertheless, since it is not listed in the dictionary, it produces a false positive in every experiment that involves Fox. It would be quite difficult to track down all such entries in the data as neither their lexemes nor their glosses match perfectly.

The performance figures are lower also because of the usual imperfections of the gold standard. In many cases, examination of apparent false positives leads to the discovery of true cognates that are not identified as such in Hewson's dictionary. Tables 5.18 and 5.19 contains examples of such word pairs. Apart from their strong similarity of form and meaning, they exhibit many of the regular sound correspondences identified by Bloomfield [1946]. Although the ultimate judgement belongs to the Algonquian scholars, it is almost certain that all those pairs are genuine cognates.

The examples in Tables 5.18 and 5.19 are by no means an exhaustive list of all new cognate pairs that could be obtained from the analysis of COGIT's output, but rather an illustrative sample. The second column from the left contains the numbers of the corresponding cognate sets in Hewson's dictionary, if applicable. The

	Set	Lang.	Lexeme	Gloss	Similarity score phon	sem	overall
1	—	Cree Ojib.	*pīsākanāpiy* *pīššākaniyāp*	'rope, rawhide thong' 'string'	0.62	0.00	0.50
2	—	Cree Ojib.	*kaskipitākan* *kaškipitākan*	'tobacco pouch' 'pouch'	0.88	0.50	0.81
3	3087	Cree Ojib.	*sīkwan* *šīkwan*	'grate for wood' 'grindstone'	0.83	0.00	0.67
4	—	Cree Ojib.	*pīsimohkān* *pīssimokkān*	'watch, clock' 'clock'	0.72	1.00	0.77
5	2897	Cree Ojib.	*pīminikan* *pīminikan*	'twist tobacco' 'auger, screw'	0.91	0.00	0.73
6	1731	Cree Ojib.	*amiskomin* *miskomin*	'yellow blackberry' 'raspberry'	0.82	0.00	0.66
7	3114	Cree Ojib.	*sīwitākan* *šīwittākan*	'salt' 'salt'	0.75	1.00	0.80
8	3570	Cree Fox	*watōw* *atōwa*	'clot of blood' 'blood-clot'	0.68	0.50	0.64
9	2665	Cree Men.	*pipon* *pepōn*	'winter' 'year, winter'	0.77	1.00	0.82
10	3114	Cree Men.	*sīwitākan* *sēwehtākan*	'salt' 'salt'	0.74	1.00	0.79
11	0843	Cree Men.	*kāhkākiw* *kākākew*	'raven' 'raven'	0.73	1.00	0.79
12	—	Cree Men.	*tōhtōsāpoy* *tōtōhsapoh*	'milk' 'milk'	0.65	1.00	0.72
13	1874	Cree Men.	*misisāhk* *mesāsāh*	'horsefly' 'horsefly'	0.64	1.00	0.71
14	0605	Cree Men.	*ēmihkwān* *ɛ̄meskwan*	'spoon' 'ladle, spoon'	0.62	1.00	0.70
15	1112	Cree Men.	*kisisōwin* *kesīswan*	'bodily heat, fever' 'fever'	0.62	1.00	0.70
16	—	Cree Men.	*sōminis* *sōmen*	'raisin' 'raisin, grape'	0.62	1.00	0.70
17	—	Cree Men.	*mimikwās* *mīmīkwɛ̄w*	'butterfly' 'butterfly'	0.62	1.00	0.69
18	—	Cree Men.	*mōsāpēw* *mōsāpɛ̄wew*	'unmarried man' 'bachelor, single man'	0.66	0.70	0.67
19	1948	Fox Men.	*mīkātiweni* *mīkātwan*	'fight' 'war, fighting'	0.66	0.70	0.67

Table 5.18: Examples of cognate pairs not included in Hewson's dictionary.

	Set	Lang.	Lexeme	Gloss	Similarity score phon	sem	overall
20	2704	Fox Men.	*pehkwikanākani* *nepēhkikanākon*	'ankle' 'ankle'	0.65	1.00	0.72
21	—	Fox Men.	*mīčiweni* *mīčehswan*	'food' 'food'	0.58	1.00	0.67
22	2502	Fox Men.	*pāškesikani* *pīhkesekan*	'gun' 'gun cap, dynamite'	0.64	0.50	0.61
23	—	Fox Men.	*apahkwayikāni* *apāhkiwikān*	'reed lodge' 'reed house'	0.63	0.50	0.60
24	2103	Fox Men.	*nahākanihkwēwa* *nohāhkaniahkiw*	'daughter in law' 'daughter-in-law'	0.62	0.50	0.60
25	1078	Fox Ojib.	*kehtikāni* *kittikān*	'field, farm' 'field, garden'	0.66	1.00	0.73
26	1570	Fox Ojib.	*manetōwa* *manitōns*	'spirit' 'worm, insect, spirit'	0.64	1.00	0.71
27	2255	Fox Ojib.	*nakamōni* *nakamowin*	'song' 'song'	0.63	1.00	0.71
28	—	Fox Ojib.	*kahkešēwi* *kekkišē*	'charcoal' 'charcoal'	0.49	1.00	0.60
29	—	Fox Ojib.	*kehčīpisōni* *kiččippisowin*	'belt' 'belt'	0.50	1.00	0.60
30	1590	Fox Ojib.	*atāmina* *mantāmin*	'maize-plant' 'grain of corn'	0.66	0.35	0.60
31	0129	Fox Ojib.	*ātesōhkākana* *ātissōkkān*	'sacred story' 'story or legend'	0.60	0.50	0.58
32	—	Men. Ojib.	*pepākeweyān* *papakiwiyān*	'shirt' 'shirt'	0.74	1.00	0.79
33	3513	Men. Ojib.	*wāwan* *wāwanw*	'egg' 'egg'	0.70	1.00	0.76
34	—	Men. Ojib.	*sōmenapoh* *šōmenāpō*	'wine' 'wine'	0.64	1.00	0.72
35	—	Men. Ojib.	*sūskīkahekan* *šōškwēka'ikan*	'flat iron' 'flat iron, iron'	0.64	1.00	0.71
36	1371	Men. Ojib.	*kōhtakan* *kuntākan*	'throat' 'throat'	0.62	1.00	0.69
37	2725	Men. Ojib.	*pīkipočekan* *pimipōčikan*	'plow' 'plough'	0.69	0.70	0.69
38	—	Men. Ojib.	*pēsekokasiw* *pēšikōkašī*	'horse' 'horse'	0.61	1.00	0.69

Table 5.19: Examples of cognate pairs not included in Hewson's dictionary (cont.).

phonetic, semantic and overall similarity scores are given in the rightmost columns. Some glosses have been truncated to preserve the space.

A successful detection of semantic similarity is usually due to some kind of surface similarity. The majority of pairs in Tables 5.18 and 5.19 have the maximum semantic similarity score of 1.00, thanks to the identity of at least one gloss. The score of 0.5, which also comes up with some frequency, identifies the cases where the keywords match even though the entire glosses do not. In pair 24, ‘daughter’ is identified as a keyword after the marking script splits ‘daughter-in-law’ into three words. Pair 23 is an example where marking only the head noun as keyword would not work. One could consider splitting the glosses containing the ‘or’ conjunction, such as the one in Pair 31, into two glosses; it is difficult however to handle cases like ‘making or supply of moccasins’.

The cases where WordNet has made the difference are fewer. In pair 2, ‘pouch’ is actually a hypernym of ‘tobacco pouch’, but keyword identity gets the same score as gloss hypernymy. Pair 18 illustrates the precedence of gloss synonymy (‘unmarried man’ $\simeq$ ‘bachelor’) over keyword identity (‘man’). In pair 37, WordNet comes to the rescue after string matching fails to associate two spelling variants. A real flash of artificial intelligence occurs in pair 30, where the semantic similarity of ‘maize-plant’ and ‘grain of corn’ is detected.

The similarity of pairs 1 and 6 was recognized by the early versions of COGIT, which computed the link distance between concepts. Both ‘rope’/‘string’ and ‘blackberry’/‘raspberry’ are *coordinate terms*, that is, words that have the same hypernym. Such a close relationship is still possible to detect using the current approach: in addition to comparing a gloss with the hypernymy list of the opposite gloss, the program would have to compare both hypernymy lists with each other. However, the real problem is that by increasing the radius of the semantic radar the program lets in an avalanche of false positives (cf. Table 5.9 on page 82). My experiments indicate that, at least until better lexical resources become available, limiting the range of comparisons to the most immediate WordNet neighbours may be the wisest strategy.

5.6 Conclusion

The results show that it is possible to identify a large portion of cognates in related languages solely on the basis of phonetic and semantic similarity. Phonetic similarity is a more reliable indication of cognation than semantic similarity. ALINE, a phonetically-based program, outperforms the simple orthographic methods on the cognate identification task. Analysis of semantic information extracted from glosses yields a dramatic increase in the number of identified cognates. Most of the gain comes from detecting entries that have matching glosses, but the use of a lexical resource can further enhance the contribution of the semantic module.

A system such as COGIT can serve as one of the principal modules of a language reconstruction system. On its own, it could be of assistance to comparative linguists dealing with large vocabulary data from languages with which they are unfamiliar. Beyond diachronic phonology, the techniques and findings presented here may also be applicable to tasks such as bitext alignment or automatic construction of translation lexicons.

Chapter 6

Determination of correspondences

The main objection that can be raised against the algorithm for cognate identification described in the previous chapter is that the phonetic alignment and the phonetic similarity score depend solely on absolute phonetic similarity. Most linguists believe that recurrent sound correspondences (henceforth referred to simply as correspondences) provide a more reliable evidence of cognation. For example, the English verb *have* is not cognate with Latin *habere* 'to have', as implied by the phonetic and semantic similarity, but rather with *capire* 'to catch'. This follows from the well-known Grimm's Law, which specifies that English [h] regularly corresponds to Latin [k]. Other known correspondences between English and Latin, such as *t:d*, *θ:t*, *n:n*, are demonstrated by the word pairs in Table 6.1. The corresponding phonemes shown in boldface originate from a single proto-phoneme. Thus, correspondences make it possible to distinguish cognates from loan words and chance resemblances. Could correspondences take the place of phonetic similarity as the basis for the identification of cognates?

The correspondences also provide decisive evidence for the relatedness of languages. However, because manual determination of correspondences is an extremely time-consuming process, it has yet to be accomplished for many proposed language

English	Latin		English	Latin	
t ɛ n	**d** e k e	'ten'	**t** ū	**d** u o	'two'
ī **t**	e **d**	'eat'	**t** ū **θ**	**d** e n **t**	'tooth'
n ɛ s **t**	**n** i **d**	'nest'	**n** ī	g e **n**	'knee'
n ɛ **f** j ū	**n** e **p** o t	'nephew'	**f** u **t**	**p** e **d**	'foot'
f ō **m**	s **p** u **m**	'foam'	w u **l** **f**	**l** u **p**	'wolf'
θ **r** ī	**t** **r** e	'three'	**r** ū **t**	**r** a **d** i k	'root'
s ɪ **t**	**s** e **d**	'sit'	**h** a **r** **t**	**k** o **r** **d**	'heart'
h ɔ **r** **n**	**k** o **r** **n**	'horn'	**b** **r** ə **ð** ə **r**	**f** **r** a **t** **r**	'brother'

Table 6.1: Examples of English–Latin cognates exhibiting correspondences.

families. A system able to perform this task automatically from unprocessed bilingual wordlists could be of great assistance to comparative linguists. The *Reconstruction Engine* of Lowe and Mazaudon [1994], a set of programs designed to be an aid in language reconstruction, as well as other approaches reviewed in Section 3.2, require a set of correspondences to be provided beforehand.

Although it may not be immediately apparent, there is a strong similarity between the task of matching phonetic segments in a pair of cognate words, and the task of matching words in two sentences that are mutual translations (Figure 6.1). The consistency with which a word in one language is translated into a word in another language is mirrored by the consistency of sound correspondences. The former is due to the semantic relation of synonymy, while the latter follows from the principle of the regularity of sound change. Thus, as already asserted by Guy [1994], it should be possible to use similar techniques for both tasks.

The primary objective of the method proposed in this chapter is the automatic determination of correspondences in bilingual wordlists, such as the one in Table 6.1. The method exploits the idea of relating correspondences in bilingual wordlists to

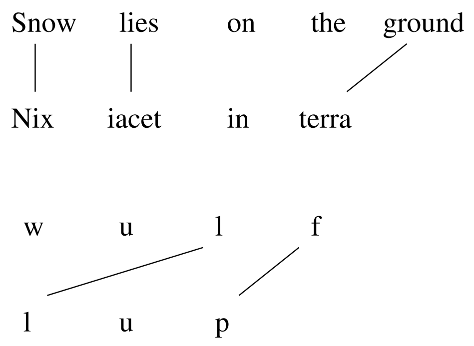

Figure 6.1: The similarity of word alignment in bitexts and phoneme alignment between cognates.

translational equivalence associations in bitexts through the employment of models developed in the context of statistical machine translation. The second task addressed in this chapter is the identification of cognates on the basis of the discovered correspondences. The experiments to be described in Section 6.5 show that the method is capable of determining correspondences in bilingual wordlists in which less than 30% of pairs are cognates, and outperforms comparable algorithms on cognate identification. Although the experiments focus on bilingual wordlists, the approach presented in this chapter could potentially be applied to other bitext-related tasks.

6.1 Related work

In a schematic description of the comparative method, the two steps that precede the determination of correspondences are the identification of cognate pairs and their phonetic alignment. Indeed, if a comprehensive set of correctly aligned cognate pairs is available, the correspondences could be extracted by simply following the alignment links. Unfortunately, in order to make reliable judgements of cognation, it is necessary to know in advance what the correspondences are. Historical linguists solve this apparent circularity by guessing a small number of likely cognates and refining the set of correspondences and cognates in an iterative fashion.

Guy [1994] outlines a correspondence-based algorithm for identifying cognates in bilingual wordlists. It is discussed here rather than in Chapter 5 because it makes no use of the notion of phonetic similarity. Since the method requires a set of word pairs large enough to support a statistical analysis, it cannot be applied to an isolated word pair.

The algorithm estimates the probability of phoneme correspondences by employing a variant of the χ^2 statistic on a contingency table, which indicates how often two phonemes co-occur in words of the same meaning. The goal of the algorithm is not to determine correspondences *per se*, but rather to use them to estimate the likelihood of cognation of word pairs. For every word pair, it finds an alignment that maximizes the sum of the correspondence probabilities, and then converts the alignment score into an estimate of cognation.

Guy's paper is not a model of clarity or rigour: the algorithm is defined by example, the theoretical grounding is shaky, and there is no quantitative evaluation on authentic language data. However, the program COGNATE, which implements the algorithm, is publicly available and can be put to a test. An experimental evaluation of COGNATE is described in Section 6.5.

Oakes [2000] adopts a much simpler approach to discovering correspondences. Sound changes are deemed to be regular if they are found to occur more than once in the aligned cognates. In contrast with Guy's method, Oakes does not compute any co-occurrence statistics, but relies solely on the phonetic similarity of words. The discovered correspondences are not used for separating cognates from non-cognates. Section 6.5 contains an evaluation of one of his programs (JAKARTA) on the cognate identification task.

Because the tasks of determination of correspondence and the identification of cognates are intertwined, some of the bitext-related algorithms implicitly determine and employ correspondences. Tiedemann [1999] considers automatic construction of weighted string similarity measures from bitexts. He includes three lists of the most frequent character "mappings" between Swedish and English, which correspond to his three mapping approaches (single characters, vowel and consonant sequences, and non-matching parts of two strings). However, because genetic cognates in the data seem to be outnumbered by borrowings, the lists contain few genuine correspondences. Mann and Yarowsky [2001] take advantage of language relatedness in order to automatically induce translation lexicons. In their search for cognates, they discover most probable character "substitutions" across languages. In the provided French–Portuguese examples, phonologically plausible correspondences *b:v, t:d* mix with mere orthographic regularities *c:q, x:s*.

Knight and Graehl [1998] in their paper on back-transliteration from the Japanese syllabic script *katakana* to the English orthography consider the sub-task of aligning the English and Japanese phonetic strings. They apply the estimation-maximization (EM) algorithm to generate symbol-mapping probabilities from 8,000 pairs of unaligned English/Japanese sound sequences. It is possible to view the sound pairs with the highest probabilities as the strongest recurrent correspondences between the two languages. Naturally, the existence of those correspondences is an artifact of the transliteration process, rather than a consequence of a genetic language relationship. Nevertheless, it may be possible to employ a similar approach to discover recurrent sound correspondences in genuine cognates. A drawback of the alignment model presented in the paper is an asymmetric, one-to-many mapping between the English and Japanese sounds, and a restricted set of edit operations that excludes both insertions and deletions. These restrictions are designed to make the models less expensive to compute.

6.2 Statistical machine translation

Statistical machine translation was proposed by the IBM group [Brown *et al.*, 1990a] as the method of building translation systems automatically from large bitexts by applying the noisy channel model. The idea is to combine a *language model*, which assigns a probability to every sentence in the target language, with a *translation model*, which assigns a probability to all pairings of the source language sentences with the target language sentences. The third main component of a statistical machine translation system, the *decoder*, actually finds the actual target language sentence that maximizes the product of the probabilities assigned by the language model and the translation model.

A translation model approximates the probability that two sentences are mutual translations by computing the product of the probabilities that each word in the target sentence is a translation of some source language word. A model of translation equivalence that determines the word translation probabilities can be *induced* from bitexts. The difficulty lies in the fact that the mapping of words in bitexts is not known in advance. The sentences and the words within the sentences have to be first *aligned* by some method. While the sentence order is by and large identical in both parts of a bitext, the word order in not. The following section discuses the problem of word alignment in bitexts in more detail.

6.2.1 The word-to-word model of translational equivalence

Algorithms for word alignment in bitexts aim at discovering word pairs that are mutual translations. Since words that are mutual translations tend to co-occur more frequently than other word pairs, a straightforward approach is to estimate the likelihood of translational equivalence by computing a similarity function based on a co-occurrence statistic, such as mutual information, Dice's coefficient, or the χ^2 test. The underlying assumption is that the association scores for different word pairs are independent of each other.

Melamed [2000] shows that the assumption of independence leads to invalid word associations, and proposes an algorithm for inducing models of translational equivalence that outperform the models that are based solely on co-occurrence counts. His models employ the so-called *one-to-one assumption*, which formalizes the observation that most words in bitexts are translated to a single word in the corresponding sentence. The algorithm, which is related to the expectation-maximization (EM) algorithm, iteratively re-estimates the *likelihood scores* which represent the probability that two word types are mutual translations.

I present here a summary of Melamed's algorithm, followed by a more detailed description of its parts. In the first step, the likelihood scores are initialized using only the co-occurrence information. Next, the likelihood scores are used to induce a set of one-to-one *links* between word tokens in the bitext. The links are determined by a greedy *competitive linking* algorithm, which proceeds to link pairs that have the highest likelihood scores. After the linking is completed, the link counts are used to re-estimate the likelihood scores, which in turn are applied to find a new set of links. The process is repeated until the translation model converges to the desired degree.

For the initialization of the likelihood scores, Melamed employs the G^2 statistic [Dunning, 1993]. The G^2 statistic provides, on the basis of a contingency table, an estimate of how unlikely it is that two tokens co-occur by chance. Let $cooc(u, v)$ be the number of co-occurrences of u and v, Furthermore,

$$k_1 = cooc(u, v), \qquad k_2 = cooc(u, \neg v) = \sum_{t \neq v} cooc(u, t)$$

$$n_1 = cooc(\cdot, v) = \sum_{s} cooc(s, v), \qquad n_2 = cooc(\cdot, \neg v) = \sum_{s} \sum_{t \neq v} cooc(s, t)$$

$$p_1 = \frac{k_1}{n_1}, \qquad p_2 = \frac{k_2}{n_2}, \qquad p = \frac{k_1 + k_2}{n_1 + n_2}$$

Then, the G^2 statistic is defined as:

$$G^2(u, v) = -2 \log \frac{B(k_1 \mid n_1, p_1) B(k_2 \mid n_2, p_2)}{B(k_1 \mid n_1, p) B(k_2 \mid n_2, p)}$$

where $B(k \mid n, p) = \binom{n}{k} p^k (1-p)^{n-k}$ denotes the probability of k being generated from a binomial distribution with parameters n and p. The pairs in which one word token is NULL are initialized to an infinitesimal value.

The greedy competitive linking algorithm is used to induce a set of links in the bitext. First, all likelihood scores are sorted in decreasing order. The pair (u, v) with the highest score is selected and all co-occurring word pairs (u, v) in the bitext are linked. For the pairs of the form $(u, NULL)$, all u word tokens are linked to NULL. Then, the process is repeated for the pair with the next highest score. Each word can be linked at most once.

Melamed presents three translation-model estimation methods. Method A re-estimates the likelihood scores as the logarithm of the probability of jointly generating the pair of words u and v:

$$score_A(u, v) = \log \frac{links(u, v)}{\sum_{u', v'} links(u', v')}$$

where $links(u, v)$ denotes the number of links induced between u and v. Note that the co-occurrence counts of u and v are not used for the re-estimation.

In Method B, an explicit noise model with auxiliary parameters λ^+ and λ^- is constructed in order to improve the estimation of likelihood scores. λ^+ is the probability that a link is induced between two co-occurring words that are mutual translations, while λ^- is the probability that a link is induced between two co-occurring words that are not mutual translations. Ideally, λ^+ should be close to 1 and λ^- should be close to 0. The actual values of the two parameters are calculated by the standard method of maximum likelihood estimation. The probability of the link frequency distributions

$$Pr(links \mid model) = \prod_{u,v} Pr(links(u,v) \mid cooc(u,v), \lambda^+, \lambda^-)$$

expressed as a function of λ^+ and λ^- has only one dominant global maximum, which can be established by hill-climbing. The *score* function is defined as:

$$score_B(u,v) = \log \frac{B(links(u,v) \mid cooc(u,v), \lambda^+)}{B(links(u,v) \mid cooc(u,v), \lambda^-)}.$$

In Method C, bitext tokens are divided into classes, such as content words, function words, punctuation, etc., with the aim of producing more accurate translation models. The auxiliary parameters are estimated separately for each class. The *score* function is defined as:

$$score_C(u, v \mid Z = class(u,v)) = \log \frac{B(links(u,v) \mid cooc(u,v), \lambda_Z^+)}{B(links(u,v) \mid cooc(u,v), \lambda_Z^-)}.$$

The evaluation was performed on a 16,000-word French–English bitext. The three translation-model estimation methods were compared to Model 1 of Brown *et al.* [1990b], which is based exclusively on co-occurrence counts. In all experiments, Methods B and C achieved significantly higher accuracy than Model 1 and Method A. When all links were considered, Method A outperformed Model 1, and Method C outperformed Method B. However, when closed-class links were ignored, Model 1 was better than Method A, and the performance of Methods B and C was roughly the same.

6.2.2 Noncompositional compounds

As a way of relaxing the *one-to-one* restriction, Melamed [1997] proposes an elegant algorithm discovering *noncompositional compounds* (NCCs) in bitexts. An NCC is a word sequence, such as "high school", whose meaning cannot be synthesized from the meaning of its components. Since many NCCs are not translated word-for-word, their detection is essential in most NLP applications.

Melamed's information-theoretic approach is based on the observation that treating NCCs as a single unit rather than as a sequence of independent words increases the predictive power of statistical translation models. Therefore, it is possible to establish whether a particular word sequence should be considered a NCC by comparing two translation models that differ only in their treatment of that word sequence. For the objective function that measures the predictive power of a translation model $Pr(s,t)$, Melamed selects *mutual information*:

$$I(S;T) = \sum_{s \in S} \sum_{t \in T} Pr(s,t) \log \frac{Pr(s,t)}{Pr(s)Pr(t)}$$

Melamed's approach to identification of NCCs is to induce a *trial translation model* that involves a candidate NCC and compare the model's total mutual information with that of a *base translation model*. The NCC is considered valid only if there is an increase of the mutual information in the trial model. The contribution of s to $I(S;T)$ is given as:

$$i(s) = \sum_{t \in T} Pr(s,t) \log \frac{Pr(s,t)}{Pr(s)Pr(t)}.$$

In order to make this procedure more efficient, Melamed proposes inducing the translation model for many candidate NCCs at the same time.

A complex gain-estimation method is used to guess whether a candidate NCC is useful *before* inducing a translation model that involves this NCC. Each candidate

NCC xy causes the net change Δ_{xy} in the objective function, which can be expressed as:

$$\Delta_{xy} = i'(x) + i'(y) + i'(xy) - i(x) - i(y),$$

where i and i' are predictive value functions for source words in the base translation model and in the trial translation model, respectively. $i'(x)$ is estimated on the assumption that the links involving x will not change in the trial translation model unless y occurs to the right of x:

$$i'(x) = i(x : RC \neq y),$$

where $(x : RC \neq y)$ denotes the set of tokens of x whose right context is y. Similarly,

$$i'(y) = i(y : LC \neq x),$$

where LC denotes word context to the left. Finally, i'(xy) is estimated as follows:

$$i'(xy) = i(x : RC = y) + i(y : LC = x).$$

Given parallel texts E and F, the algorithm iteratively augments the list of NCCs. The iteration starts by inducing a base translation model between E and F. All continuous bigrams which are estimated to increase mutual information of the translation model are placed on a sorted list of candidate NCCs, but for each word token, only the most promising NCC is allowed to remain on the list. Next, a trial translation model is induced between E' and F, where E' is obtained from E by fusing each candidate NCC into a single token. If the net change in mutual information gain contributed by a candidate NCC is greater than zero, all occurrences of that NCC in E are permanently fused; otherwise the candidate NCC is placed on a stop-list. The entire iteration is repeated until reaching an application-dependent stopping condition.

The method was evaluated on a large English–French bitext containing transcripts of Canadian parliamentary debates (Hansards). In one experiment, after six iterations the algorithm identified on both sides of the bitext about four hundred NCCs that increased the mutual information of the model. Another experiment, which is particularly relevant for the application discussed in this chapter, showed that the method was capable of discovering meaningful NCCs in a data set consisting of spellings and pronunciations of English words (for example, "ng" was determined to be a NCC of English spelling because it consistently "translates" into the sound /ŋ/). Nevertheless, the author admits that the full NCC recognition algorithm was not tested in any real application.

6.3 Models of phoneme correspondence

Thanks to its generality, Melamed's parameter estimation process can be adapted to the problem of determining correspondences. Moreover, his models are symmetric, which makes them more suitable for this task than the related IBM models [Brown *et al.*, 1990a]. The main idea is to induce a model of sound correspondence in a bilingual wordlist, in the same way as one induces a model of translational equivalence among words in a parallel corpus. After the model has converged, phoneme pairs with the highest likelihood scores represent the most likely correspondences.

The most important modification of the original algorithm concerns the method of aligning the segments in two corresponding strings. In sentence translation, the links frequently cross and it is not unusual for two words in different parts of sentences to correspond. On the other hand, the processes that lead to link intersection in diachronic phonology, such as *metathesis*, are sporadic. By imposing the no-crossing-links constraint on alignments, a dramatic reduction of the search space is achieved, and the approximate *competitive linking algorithm* of Melamed can be replaced with a variant of the well-known dynamic programming algorithm of Wagner and Fisher [1974].

Null links in statistical machine translation are induced for words on one side of the bitext that have no clear counterparts on the other side of the bitext. Melamed's algorithm explicitly calculates the likelihood scores of null links for every word type occurring in a bitext. In diachronic phonology, phonological processes that cause insertion or deletion of segments usually operate on individual words rather than across the language. Therefore, I model insertion and deletion by employing an *indel* penalty for unlinked segments.

The alignment score is computed by summing the number of induced links, and applying a small constant penalty for each unlinked segment, with the exception of the segments beyond the rightmost link (the so-called *half-local alignment* described in Section 4.4.3). The exception reflects the relative instability of word endings in the course of linguistic evolution. Metathesis is not considered. In order to avoid inducing links that are unlikely to represent recurrent sound correspondences, only pairs whose likelihood scores exceed a set threshold are linked. All correspondences above the threshold are considered to be equally valid. In the cases where more than one best alignment is found, each link is assigned a weight that is its average over the entire set of best alignments (for example, a link present in only one of two competing alignments receives the weight of 0.5).

The NCC algorithm is adapted with one major change. After inducing a trial translation model between E' and F, the original algorithm accepts all candidate NCCs that contribute a positive net change in mutual information gain. For the detection of phoneme NCCs, I decided to accept all candidate NCCs that result in a correspondence that has a likelihood score above the minimum-strength threshold t described in the following section. I found that the strength of an induced correspondence better reflects the importance of a phoneme cluster than the mutual information gain criterion.

6.4 Implementation

The method described above has been implemented as a C++ program, named CORDI, which takes as input a bilingual wordlist and produces an ordered list of correspondences. A model for a 200-pair list usually converges after 3–5 iterations, which takes only a few seconds on a Sparc workstation. The user can choose between methods A, B, and C, described in Section 6.2, and an additional Method D. In Method C, phonemes are divided into two classes: non-syllabic (consonants and glides), and syllabic (vowels); links between phonemes belonging to different classes are not induced. Method D differs from Method C in that the syllabic phonemes do not participate in any links.

Adjustable parameters include the indel penalty ratio d and the minimum-strength correspondence threshold t. The parameter d fixes the ratio between the negative indel weight and the positive weight assigned to every induced link. A lower ratio causes the program to be more adventurous in positing sparse links. The parameter t controls the tradeoff between reliability and the number of links. In Method A, the value of t is the minimum number of links that have to be induced for the correspondence to be valid. In methods B, C, and D, the value of t implies a score threshold of $t \cdot \log \frac{\lambda^+}{\lambda^-}$, which is a score achieved by a pair of phonemes that have t links out of t co-occurrences. In all experiments described below, d was set to 0.15, and t was set to 1 (sufficient to reject all non-recurring correspondences). In Method D, where the lack of vowel links causes the linking constraints to be weaker, a higher value of $t = 3$ was used.

The NCC approach is activated by specifying a -nN option, where N is the maximum number of iterations of the algorithm. The algorithm may terminate sooner if two subsequent iterations fail to produce any candidate NCCs.

6.5 Evaluation

This section describes four experiments aimed at assessing CORDI's accuracy.

6.5.1 The data for experiments

The raw vocabulary lists and the dictionary of Proto-Algonquian described in Section 5.4.1 constitute a valuable test data set also in this chapter. The Algonquian correspondences are not included in the set, but they are relatively well documented in [Bloomfield, 1925; 1946]. Because of the large number of complex 1:2 and 2:2 correspondences, the Algonquian languages are ideal for testing the NCC approach.

The Comparative Indo-European Data Corpus is an important lexical resource that is freely available on the World Wide Web. The corpus contains the Swadesh 200 lists for over 80 Indo-European languages and dialects, together with almost complete cognation judgements. It was compiled in the 1960s by the renowned historical linguist Isidore Dyen of Yale University, and played an important part in lexicostatistical research [Dyen *et al.*, 1992]. The data was originally represented on punched cards by 26 ASCII letters, with diacritical marks handwritten on the cards. Unfortunately, as the diacritical marks have not yet been transferred to electronic form, the words are unsuitable for automatic phonetic analysis.

Kessler [2001] includes in his book the Swadesh 200 lists for Albanian, English, French, German, Latin, Hawaiian, Navajo, and Turkish. The first five languages belong to the Indo-European family and share a number of cognates. The lists are available in the XML format from the author's web page. The data is of high quality.

Four variants of each etymon are provided: the root and the stem are given in a phonetic transcription, and the full word is specified both in the phonetic and in the orthographic form. The cognation judgements are specified either as incontrovertible or as doubtful. Succinct notes provide bibliographical references as well as other information.

In order to go beyond the Swadesh list of basic concepts, I extracted from the linguistic literature a set of 112 established English–Latin cognate pairs. English words were transcribed into phonetic notation with the aid of the publicly available Carnegie Mellon Pronouncing Dictionary, version 0.6. A straightforward *Perl* script was sufficient to map letters to sounds in Latin words. The entire set is listed in Appendix E.

6.5.2 Determination of correspondences in cognate pairs

In order to test CORDI's ability to determine correspondences in clean cognate data, Method D was applied to the set of 112 English–Latin cognate pairs given in Appendix E. The correspondences with scores above the threshold of $t = 3$ are shown in Table 6.2. The perfect agreement between the number of links and the number of co-occurrences is striking, but not implausible, considering that the input set includes many "textbook" examples of cognates, which are usually selected to illustrate the regularity of correspondences. CORDI correctly identifies the most salient segments that have been preserved from Proto-Indo-European, and partially re-discovers the venerable Grimm's Law, a set of sound changes that occurred in prehistoric times in Proto-Germanic. As a result of discovering the *t:d* and *θ:t* correspondences, the program correctly aligns the troublesome *tooth:dente* pair, which was beyond the power of a phonetic aligner (Section 4.8.1). The erroneous *y:w* correspondence is caused by

	cooc	*links*	*score*	valid
r:r	28	28	193.1	yes
n:n	23	23	158.6	yes
l:l	20	20	138.0	yes
s:s	17	17	117.3	yes
m:m	15	15	103.5	yes
f:p	13	13	89.7	yes†
t:d	11	11	75.9	yes†
k:g	8	8	55.1	yes†
y:w	6	6	41.4	no
b:f	6	6	41.4	yes†
h:k	5	5	34.5	yes†
θ:t	4	4	27.6	yes†

Table 6.2: English–Latin correspondences discovered by Method D in pure cognate data. The correspondences marked with a † are predicted by Grimm's Law.

the practice of transcribing the English diphthong [aɪ] as a vowel-consonant pair `ay`.

For comparison, Table 6.3 contains the correspondences identified by Oakes's program JAKARTA program in the same data. Only the correspondences identified more than three times are shown. JAKARTA is clearly less effective in linking the related segments. For example, only 12 out of 20 co-occurrences of /l/ are detected, whereas CORDI classifies all 20 co-occurrences as correspondences. A quick perusal of the relevant cognates confirms that this is indeed the case. It is not surprising, therefore, that overall CORDI discovers more correspondences than JAKARTA.

6.5.3 Determination of correspondences in word pairs

The assumption that a set of identified cognates is already available as the input for the program is not very plausible. The very existence of a reliable set of cognate pairs implies that the languages in question have already been thoroughly analyzed and that the sound correspondences are known. A more realistic input requirement is a list of word pairs from two languages where that the corresponding words have the

RSC	*cooc*	*count*	valid
r:r	28	21	yes
n:n	23	17	yes
s:s	17	17	yes
m:m	15	14	yes
l:l	20	12	yes
f:p	13	11	yes
t:t	8	8	yes
b:f	6	5	yes
k:g	8	4	yes
y:k	6	4	no

Table 6.3: Correspondences discovered by JAKARTA in pure cognate data.

same, well-defined meaning (such as the bilingual wordlist in Table 2.3 on page 11). Determining correspondences in a list of synonyms is clearly a more challenging task than extracting them from a list of reliable cognates because the non-cognate pairs introduce noise into the data. Note that Melamed's original algorithm is designed to operate on aligned sentences that are mutual translations.

In order to test CORDI's ability to determine correspondences in noisy data, Method D was applied to Kessler's English–Latin 200-word list. Only 29% of word pairs are actually cognate; the remaining 71% of the pairs are unrelated lexemes. The top ten correspondences discovered by the program are shown in Table 6.4. Remarkably, all but one are valid.

For comparison, the top ten phoneme matchings picked up by the χ^2 statistic are shown in Table 6.5. Only four of them are valid correspondences. $freq_1$ and $freq_2$ refer to the frequencies of the pertinent phonemes in the English and Latin data, respectively. Macrons denote double consonants, which are transcribed as single segments in Kessler's data. One of the weaknesses of the χ^2 statistic is illustrated by the fact that four of the pairs in Table 6.5 have high χ^2 values in spite of a single co-occurrence in the data. It is tempting to try to correct this bias by establishing

	cooc	*links*	*score*	valid
r:r	26	24	158.7	yes
n:n	24	23	154.2	yes
t:d	18	18	122.4	yes
k:k	12	11	72.5	yes
s:s	11	10	65.7	yes
f:p	9	9	61.2	yes
m:m	10	9	58.9	yes
d:t	10	8	49.8	no
l:l	14	9	49.7	yes
h:k	7	7	47.6	yes

Table 6.4: English–Latin correspondences discovered by CORDI in noisy synonym data.

	$freq_1$	$freq_2$	*cooc*	χ^2	valid
θ:s̄	10	1	1	22.3	no
θ:r̄	10	1	1	22.3	no
n:n	43	60	24	14.4	yes
m:m	20	30	10	12.9	yes
f:p	24	24	9	11.6	yes
t:d	49	37	18	11.6	yes
l:f	37	15	9	10.8	no
ð:n̄	10	2	1	9.0	no
ŋ:g	6	21	3	8.7	no
v:y	6	3	1	8.4	no

Table 6.5: English–Latin correspondences discovered using the χ^2 statistic.

a threshold for the number of co-occurrences. Guy's [1994] program COGNATE contains a number of such *ad hoc* corrections. The following section includes an assessment of the program's effectiveness.

6.5.4 Identification of cognates in word pairs

The quality of correspondences produced by CORDI is difficult to validate, quantify, and compare with alternative approaches. However, it is possible to evaluate the correspondences indirectly by using them to identify cognates. The likelihood of cognation of a pair of words increases with the number of correspondences that they contain. Since CORDI explicitly posits correspondence links between words, the likelihood of cognation can be estimated by simply dividing the number of induced links by the length of the words that are being compared. A minimum-length parameter can be set in order to avoid computing cognation estimates for very short words, which tend to be unreliable. However, in order to provide a fair comparison with the programs that do not impose any length constraints on words, the minimum-length parameter was not used in the experiments described here.

r_i	word pair	cognate?	i	p_i
1	/hɑrt/:/kord/	yes	1	1.00
2	/hɑt/:/kalid/	no		
3	/snō/:/niw/	yes	2	0.66

Table 6.6: An example ranking of cognate pairs.

The evaluation method for cognate identification algorithms adopted in this section is to apply them to a bilingual wordlist and order the pairs according to their scores (refer to Table 6.6). The ranking is then evaluated against a gold standard by computing the n-point average precision, where n is the total number of cognate pairs in the list. The n-point average precision is obtained by taking the average of n precision values.

$$P = \frac{\sum_{i=1}^{n} p_i}{n}$$

The partial precision values p_i are calculated for each point in the list where we find a cognate pair by the following formula:

$$p_i = \frac{i}{r_i}$$

where i is the number of the cognate pair counting from the top of the list produced by the algorithm, and r_i is the rank of this cognate pair among all word pairs. The

n-point precision of the ranking in Table 6.6 $(1.0+0.66)/2 = 0.83$. A perfect ordering of all cognate pairs before all non-cognate pairs achieves $P = 1.0$. The expected n-point precision of a program that randomly orders word pairs is close to c, where c is the proportion of cognate pairs in the list.

The n-point precision can further be normalized as: $\kappa = \frac{P-c}{1-c}$. Such a normalization allows to compare accuracy without the need to take into account the density of cognates in the input word list. The maximum value of κ is of course 1.0. The expected value of κ for a random ordering turns out to be independent of c, and approaches zero as the number of pairs N increases:

$$\kappa(N) = (H_N - 1)/(N - 1)$$

where H_n is the n-th harmonic number. $\kappa(200)$ is approximately .0245.

The development set consisted of three Swadesh 200 list pairs adapted from the Comparative Indo-European Data Corpus. The Polish–Russian, Spanish–Rumanian, and Italian–Serbo-Croatian were selected because they represent three different levels of relatedness, and also because they have relatively transparent grapheme-to-phoneme conversion rules. They were transcribed into a phonetic notation by means of *Perl* scripts and then stemmed and corrected manually. The ambiguities caused by the imperfect transcription were corrected with the help of dictionaries and/or native speakers of the languages in question.

Table 6.7 compares the average precision achieved by methods A, B, C, and D on the development set. The cognation judgements from the Comparative Indo-European Data Corpus served as the gold standard. The 'Cognates' column contains the ratio of the number of cognate pairs to the length of the list. Naturally, the smaller the ratio, the harder the task becomes.

In order to objectively compare the methods proposed in this paper with other

Languages		Proportion of cognates	Method A	B	C	D
Polish	Russian	.735	.989	.994	.994	.986
Rumanian	Spanish	.585	.898	.948	.948	.875
Italian	Serbo-Croatian	.253	.499	.455	.527	.615

Table 6.7: Average cognate identification precision on the development set for various methods.

cognate identification programs, five 200 Swadesh lists compiled by Kessler [2001] were adopted as the test set. As the lists contain rich phonetic and morphological information, the stemmed forms were automatically converted from the XML format with virtually no extra processing. The word pairs classified as doubtful cognates were assumed to be unrelated.

The lists represent five Indo-European languages: English, German, French, Latin, and Albanian. Apart from the English–German and the French–Latin pairs, all remaining language pairs are quite challenging for a cognate identification program. In many cases, the gold-standard cognate judgments distill the findings of decades of linguistic research. In fact, for some of those pairs, Kessler finds it difficult to show by statistical techniques that the surface regularities are unlikely to be due to chance. Nevertheless, in order to avoid making subjective choices, CORDI was evaluated on all possible language pairs in Kessler's set.

Two programs mentioned in Section 6.1, COGNATE and JAKARTA, were also applied to the test set. COGNATE is available as an MS-DOS executable on the World Wide Web. A special *Perl* script was written to transcribe the phonetic input data into the required one-letter-per-phoneme format. Unfortunately, the program's highly interactive nature (Table 6.8) makes it difficult to analyze the results of pro-

```
    p   i   s   k   i
f  82  33  55  59  33
I  41  70  38  62  70
S   0  63   0  49  63


fIS/piski  (word pair #49, pass #1)
I am 28% sure that they are NOT related.
I allowed only for matches >= 50.
The best I found were: fI  S
                       piski

N(ext), P(revious), J(ump), M(inimum match), O(ther pass), Q(uit) ?
```

Table 6.8: A sample screen from Guy's program COGNATE.

cessing entire wordlists. The problem is aggravated by the fact that the MS-DOS operating system is no longer in common use. For each word pair, the output had to be manually captured and pasted into a file, which was subsequently processed by another dedicated *Perl* script. The script ordered the word pairs according to the decreasing confidence scores, with the ties broken randomly. It took several thousands of mouse clicks to arrive at the average precision figures for the ten test wordlists.

The source code of JAKARTA was obtained directly from the author. Since in its original form it only recognizes 26 phonemes, the program was amended according to the author's instructions in order to make it recognize additional phonemes. As in the case of COGNATE, special scripts handled the input data conversion and the calculation of the average precision. Word pairs were ordered according to the increasing edit distances, with random breaking of ties. Because of the adopted evaluation method, JAKARTA's edit distance threshold of 4 for separating cognates from non-cognates was ignored.

The results on the test set are shown in Table 6.9. The best result for each language pair is underlined. The performance of COGNATE and JAKARTA is quite similar, even though they represent two radically different approaches to cognate identification. On average, methods B, C, and D outperform both comparison programs. On closely related languages, Method B, with its relatively unconstrained linking, achieves the highest precision. Method D, which considers only consonants, is the best on fairly remote languages, where vowel correspondences tend to be weak. The only exception is the extremely difficult Albanian–English pair, where the relative ordering of methods seems to be accidental. As expected, Method A is outperformed by methods that employ an explicit noise model. However, in spite of its extra complexity, Method C is not consistently better than Method B, perhaps because of its inability to detect important vowel-consonant correspondences, such as the ones between French nasal vowels and Latin /n/. The results of Table 6.9 are normalized as κ in Table 6.10.

6.5.5 Determination of complex correspondences

In all the experiments described so far, the correspondences were assumed to consist of single phonemes. In order to test the suitability of the NCC approach, a separate experiment was performed on a subset of the Algonquian data. As in Section 6.5.3, the goal was to determine correspondences from noisy wordlists. In this experiment, it was possible to rigorously evaluate the resulting list of correspondences by comparing it to the set of correspondences determined by Bloomfield [1925; 1946].

The input data was automatically extracted from the raw vocabulary lists by selecting all pairs of noun lexemes that had at least one gloss in common. The end result of such an operation is bilingual wordlists containing both cognate and non-

Languages		Proportion of cognates	COGNATE	JAKARTA	Method			
					A	B	C	D
English	German	.590	.878	.888	.936	.957	.952	.950
French	Latin	.560	.867	.787	.843	.914	.838	.866
English	Latin	.290	.590	.447	.584	.641	.749	.853
German	Latin	.290	.532	.518	.617	.723	.736	.857
English	French	.275	.324	.411	.482	.528	.545	.559
French	German	.245	.390	.406	.347	.502	.487	.528
Albanian	Latin	.195	.449	.455	.403	.432	.568	.606
Albanian	French	.165	.306	.432	.249	.292	.319	.437
Albanian	German	.125	.277	.248	.156	.177	.154	.312
Albanian	English	.100	.225	.227	.302	.373	.319	.196
Average		**.283**	**.484**	**.482**	**.492**	**.554**	**.567**	**.616**

Table 6.9: Average cognate identification precision on the test set for various methods.

Languages		Proportion of cognates	COGNATE	JAKARTA	Method			
					A	B	C	D
English	German	.590	.702	.726	.844	.894	.882	.877
French	Latin	.560	.697	.517	.643	.804	.633	.695
English	Latin	.290	.423	.221	.415	.494	.646	.794
German	Latin	.290	.341	.321	.461	.610	.628	.799
English	French	.275	.067	.188	.285	.349	.372	.392
French	German	.245	.192	.213	.136	.340	.320	.375
Albanian	Latin	.195	.315	.322	.258	.294	.463	.511
Albanian	French	.165	.169	.320	.101	.153	.185	.325
Albanian	German	.125	.174	.141	.035	.059	.033	.214
Albanian	English	.100	.139	.141	.225	.304	.244	.107
Average		**.283**	**.322**	**.311**	**.340**	**.430**	**.441**	**.509**

Table 6.10: Average cognate identification precision normalized as κ.

cognate pairs. This method of producing bilingual wordlists is not only faster than the manual compilation, but also has the potential of incorporating cognates that are outside the basic set of 200 concepts.

The Cree–Ojibwa list served as the development set, and the Fox–Menomini list as the test set. Again, the latter turned out to be more challenging than the former. The Cree–Ojibwa contained 732 pairs, including 242 (33.1%) cognate pairs. The Fox–Menomini list contained 397 word pairs, including only 79 (19.9%) cognate pairs. This is less than the number given in Table 5.12 on page 85 because not all cognate pairs have a gloss in common.

Since the vowel correspondences in Algonquian are rather inconsistent, following Hewson [1974], I decided to concentrate on consonants and consonant clusters. The models were induced using Method C, which separates the syllabic and the non-syllabic phonemes. On the Fox–Menomini data, the algorithm terminated after 12 iterations, which took several minutes. (When the NCC option is switched on, the computation takes much longer because each iteration involves inducing anew both the base and the trial translation models.)

Figure 6.2 shows the correspondences determined by CORDI (the right circle) intersected with the ones enumerated by Bloomfield (the left circle). S represents the phoneme [ʃ], C represents [ʧ], and q represents the glottal stop [ʔ]. The intersection of both circles contains 20 correspondences that were correctly identified by the program. The three erroneous correspondences on the right (false positives) can be traced to alignments of unrelated words. The resulting precision was therefore 87%. As for the false negatives on the left, I manually analyzed the 79 cognate pairs available to the program, and found that *š:hk* and *p:hp* occur twice, *hč:qč* occurs once, and the remaining seven complex correspondences do not occur at all. The *h:q* correspondence is dubious because it only occurs within clusters. Since, by definition, recurrent

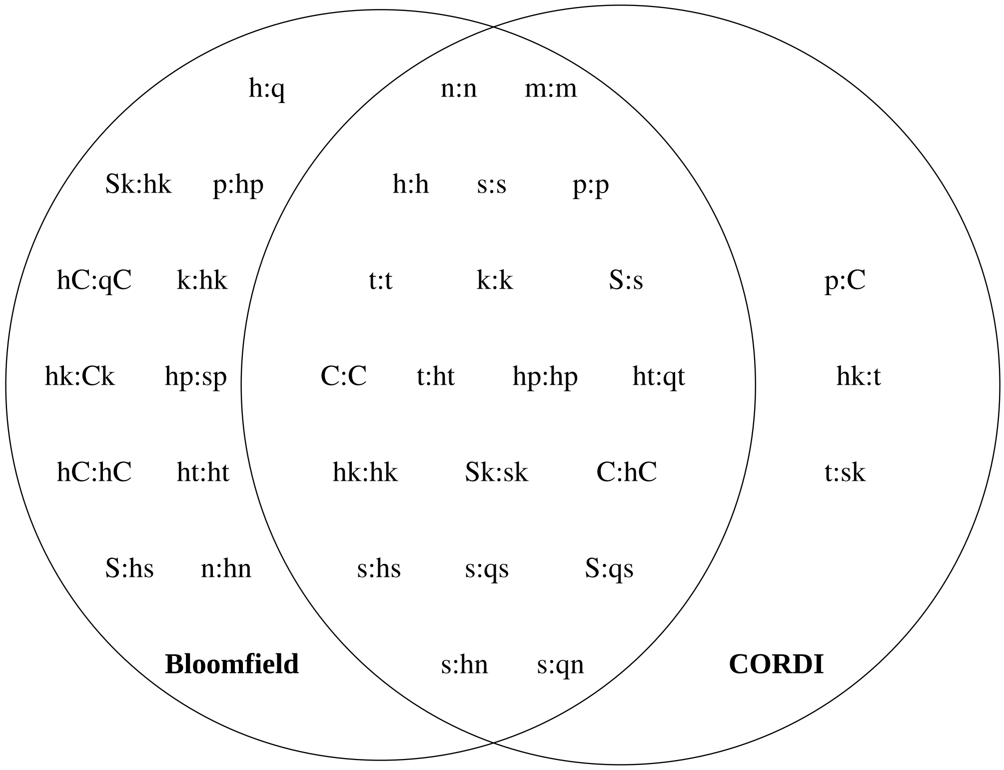

Figure 6.2: The Fox–Menomini consonantal correspondences determined by a linguist and by CORDI.

correspondences are those that occur at least twice, the recall on the test set was 91%. For comparison, on the same Fox–Menomini list, JAKARTA identifies only eight consonantal correspondences: *n:n, m:m, h:h, s:s, p:p, t:t, k:k,* and *h:hs*, of which the single complex correspondence is not in Bloomfield's set.

The results of the experiment are extremely encouraging. The accomplishment of a very high precision *and* recall on a test set composed of 80% noise confirms that the iterative statistical approach advocated here is highly robust. The impressive outcome should, however, be interpreted with caution. Because of the (unavoidably) small number of target correspondences, the change of a single classification makes a difference of about 5% in the resulting precision/recall figures. Moreover, the decision to ignore vowels and glides helped the program to focus on the right type of correspondences. Finally, the Algonquian consonantal correspondences are almost context-free, which nicely suits the program's principles. Nevertheless, it is a very satisfying situation when the author has to justify the results that seem too outstanding.

6.6 Conclusion

I have presented a novel approach to the determination of correspondences in bilingual wordlists. The results of experiments indicate that the approach is robust enough to handle a substantial amount of noise that is introduced by unrelated word pairs. CORDI does well even when the number of non-cognate pairs is more than double the number of cognate pairs. When tested on the cognate-identification task, CORDI achieves substantially higher precision than comparable programs.

Since the correspondences determined by my program are explicitly posited, they can be verified by examining individual cognate pairs. This is in contrast with statistical approaches of Ringe [1998] and Kessler [2001] that estimate the probability of the existence of cognates in bilingual wordlists. Those methods do not provide verifiable evidence beyond a single value which expresses the likelihood that the correlation between languages is statistically significant. In addition, due to the stochastic nature of the methods, such results are often difficult to reproduce. In my opinion, the significance tests alone are unlikely to convince historical linguists.

Another advantage of the algorithm presented here is its capability of linking phonemes in any word position. The approaches that rely on the syllabic structure of words [Ringe, 1992] or the character position within word [Tiedemann, 1999] tend to produce rigid alignments, which are unable to handle phenomena such as epenthesis (insertion of a vowel between consonants) or syncope (loss of a vowel between consonants).

The results presented here prove that the techniques developed in the context of statistical machine translation can be successfully applied to a problem in diachronic phonology. I am convinced that the transfer of methods and insights is also possible in the other direction.

Chapter 7

Wrap-up and outlook

I have presented algorithms for the three principal steps of the comparative method of language reconstruction: the alignment of cognates, the identification of cognates, and the determination of correspondences. In this final chapter, I discuss the possibility of integrating all three components into a self-contained program capable of performing a large part of the reconstruction process, and I point out some other directions for future research. I conclude with a review of the main results presented in the thesis.

7.1 Identification of cognates

In this thesis, I have presented two distinct algorithms for the identification of cognates: a similarity-based method described in Chapter 5, and a correspondence-based method introduced in Chapter 6. Both approaches are valid and have been used before: Oakes's program JAKARTA is an example of the former, while Guy's COGNATE belongs to the latter. Similarly, for the related task of estimating the likelihood of historical connection between languages, Baxter and Manaster Ramer [2000] and Oswalt [1998] employ similarity-based measures, while Ringe [1998] and Kessler [2001]

Languages		JAKARTA	Trunc.	DICE	LCSR	ALINE
English	German	.888	.856	.786	.903	.918
French	Latin	.787	.792	.653	.809	.865
English	Latin	.447	.515	.360	.539	.733
German	Latin	.518	.443	.399	.498	.704
English	French	.411	.570	.428	.561	.616
French	German	.406	.435	.382	.483	.497
Albanian	Latin	.455	.558	.437	.582	.623
Albanian	French	.432	.595	.451	.487	.613
Albanian	German	.248	.253	.310	.282	.307
Albanian	English	.227	.151	.194	.181	.276
Average		**.482**	**.517**	**.440**	**.533**	**.615**

Table 7.1: Average cognate identification precision on the test set for various similarity-based methods.

concentrate on correspondences. Is it possible to determine which of the two approaches achieves better results?

Although it would be difficult to settle this issue once and for all, it is relatively straightforward to compare the algorithms proposed in this thesis by calculating their average precision on the same input data. Table 7.1 shows the performance of various similarity-based methods on five 200 Swadesh lists compiled by Kessler [2001]. Because the words in the lists are already matched by meaning, the semantic component of COGIT is not applicable here, which means that the identification of cognates is performed solely on the basis of phonetic similarity. ALINE's parameter settings are the same as used in Chapter 4. The results confirm again that ALINE performs consistently better than other methods. Surprisingly, some orthographic measures outperform Oakes's program JAKARTA.

In order to facilitate comparison with the correspondence-based methods, Ta-

Languages		Cognates	Method B	Method D	ALINE
English	German	.590	.957	.950	.918
French	Latin	.560	.914	.866	.865
English	Latin	.290	.641	.853	.733
German	Latin	.290	.723	.857	.702
English	French	.275	.528	.559	.615
French	German	.245	.502	.528	.497
Albanian	Latin	.195	.432	.606	.622
Albanian	French	.165	.292	.437	.612
Albanian	German	.125	.177	.312	.307
Albanian	English	.100	.373	.196	.276
Average		**.283**	**.554**	**.616**	**.615**

Table 7.2: Average cognate identification precision on the test set for various methods.

ble 7.2 reproduces some of the results from Chapter 6. On average, ALINE does about as well as Method D and better than other correspondence-based methods proposed in Section 6.5.4. Even on the language pairs involving Albanian, where the proportion of cognates is below 20%, ALINE's performance is surprisingly good.

Both the correspondence-based and the similarity-based methods perform well. Intuitively, a judicious combination of both techniques should produce even better results. One possible approach is to use the information about correspondences to *boost* the similarity values of certain phoneme pairs. First, the translation model would be induced on the data as described in Chapter 6. The phoneme pairs representing strong correspondences (correspondences that have a likelihood score above the threshold t) would have their similarity scores increased by a constant value. This constant value should be a parameter that can be established on a development set. A design problem that must be solved in order to implement this idea is the incorporation of complex, multi-phoneme correspondences into a similarity scheme that is currently defined only for pairs of individual phonemes.

The combined similarity-cum-correspondences approach could also be used for the task of identifying cognates from raw vocabulary data, in the manner described in Chapter 5. The idea is to first identify a set of likely cognate pairs, then induce a translation model on the set, and finally use the determined correspondences to improve COGIT, the cognate identification algorithm. Such an approach would integrate virtually all algorithms proposed in this thesis.

One possible way to determine the set of likely cognate pairs is to select n candidate pairs starting from the top of the ordered list produced by COGIT. After all, these are the most likely cognate pairs. The problem with such an approach is that the selected pairs are certain to exhibit high phonetic similarity. When a translation model is induced on such set, the strongest correspondences can be expected to consist mostly of pairs of identical phonemes. This is because the set of candidates is going to be *biased* by the method used for their selection.

A better idea is to select candidate cognates on the basis of semantic similarity only. Analysis of the Algonquian vocabulary lists shows that selecting all vocabulary entries characterized by the highest level of semantic similarity, that is, gloss identity, produces a set of candidate pairs containing a substantial proportion of cognates. Table 7.3 shows the number of pairs that have at least one gloss in common, and the proportion of cognates in such set. The experiments described in Section 6.5.5 indicate that such levels of cognate content may be sufficient for the determination of the strongest correspondences. Again, the values of the boosting constant and the mixing parameter α should be estimated on a development set.

Languages		All pairs	Cognate pairs	Proportion of cognates
Fox	Menomini	397	79	.199
Fox	Cree	409	85	.208
Fox	Ojibwa	326	86	.263
Menomini	Cree	834	146	.175
Menomini	Ojibwa	656	160	.244
Cree	Ojibwa	732	242	.331

Table 7.3: The proportion of cognates among the pairs that have at least one gloss in common.

7.2 Reconstruction

Following the determination of correspondences, the next step of the comparative method is the reconstruction of proto-phonemes and proto-forms. For each correspondence, the linguist posits a phonetic segment as the hypothetical source of the pair of phonemes that make up the correspondence. In the previous chapters, I have proposed algorithms for the antecedent stages of the reconstruction process. Is it possible to automate also this final step of the comparative method?

The rules of comparative reconstruction are intuitive and informal. The selection of phonemes requires extensive knowledge about the languages in question, as well as about the typological principles that determine the plausibility of a given phonological system. In order to reconstruct, say, Proto-Germanic, the facts about other Indo-European sub-families have to be taken into account. Also, the reconstructed system should not have abnormal properties (for example, a total lack of fricatives) that are at odds with what is known about existing languages. Ideally, a proposal should include an ordered set of regular sound changes which transform the hypothesized proto-language etyma into the attested forms.

The proto-languages proposed by independent researchers are often very different. The following observation by Greenberg summarizes the current views on Proto-Indo-European.

> After more than a century of effort, which has involved the majority of the linguistic community in the nineteenth century and a substantial group of specialists in the twentieth, it is not too much to say that the only matters on which everyone agrees in regard to the sound system of Proto-Indo-European are that there were at least four points of articulation for stops, including at least labials and dentals or alveolars, at least three manners of articulation for stops, and at least one sibilant and one vowel. [Greenberg, 1993]

He then gives an example of laryngeals, of which "almost every number from zero to ten or more has been posited". There does not seem to be any accepted procedure for evaluating the validity of a hypothesized proto-language. Specific proposals regarding relatedness of languages and the shape of the genealogical trees of language families may become gradually accepted by achieving the support of the majority of linguists.

In my opinion, the design of the phonological proto-system is best left to human experts. In the future, I would like to implement a module that facilitates such a task by allowing the user to propose a proto-phoneme for each of the correspondences determined by CORDI, and subsequently generates a proto-form for each of the identified cognate sets. Such an interactive approach would have much in common with the *Reconstruction Engine* of Lowe and Mazaudon [1994], with the important difference of relieving the user of the task of determining correspondences. As an example, let us assume that the user has tagged the correspondences in Table 7.4 with the reconstructed segments shown in boxes, The program could then automatically posit Proto-Algonquian **m(e:ɛ)çk(w)i(h)* for the possible cognate pair *meškwi* (Fox, 'blood') and *mɛhkīh* (Menomini, 'blood'). The user could continue the reconstruction

Fox	Menomini	Proto-segment
m	m	$\boxed{\text{m}}$
šk	hk	$\boxed{\text{çk}}$
i	ī	$\boxed{\text{i}}$

Table 7.4: A partial table of correspondences.

by deciding on the status of segments included in brackets, perhaps ending up with something like **meçkwi*.

In the future it should be possible to integrate ALINE, COGIT, and CORDI, and the proposed reconstruction module into a system that given nothing more than vocabulary lists of two related languages outputs a partial reconstruction of the lexicon of their proto-language. Other issues that should be addressed before such a system becomes functional include the construction of a comprehensive training corpus of aligned cognates that would make it possible to to derive the similarity matrices automatically, and the extension of the algorithms proposed here to handle more than two languages at the same time.

Although it may seem somewhat disappointing to stop short of creating a computer system that performs the entire process of reconstruction of proto-languages in a fully unsupervised fashion, the algorithms proposed in this thesis are intended to assist rather than replace comparative linguists. The proposed correspondences, cognates, and their alignments should be considered as a starting point for further investigations. The information that can be gleaned from the word lists must be supplemented by other types of information, which include the morphology and syntax of the languages in question, the clues from related languages, general linguistic knowledge, and sometimes even extra-linguistic sources. As in machine translation, it would be unrealistic to expect a computer program to exceed or even match the performance of a human expert on such a difficult problem. The best results may come from combining the speed and thoroughness of the former with the intuition and knowledge of the latter.

7.3 Summary of results

In this thesis, I have reported innovative algorithms for the three principal steps of the comparative method of language reconstruction: the alignment of cognates, the identification of cognates, and the determination of recurrent sound correspondences.

In Chapter 4, I considered the problem of the alignment of phonetic strings, which is a necessary step in many applications in computational phonology. After discussing various approaches to phonetic alignment, I presented a new algorithm that combines the dynamic programming approach to finding the optimal alignment with a complex scoring scheme for computing phonetic similarity between phonetic segments. The new algorithm, which is inspired by ideas taken from bioinformatics, incorporates several techniques developed for sequence comparison: an extended set of edit operations that includes compression and expansion, local and semi-global modes of alignment, and the capability of retrieving a set of near-optimal alignments. Unlike previously proposed distance-based algorithms, it is based on the notion of similarity between sounds, which is likely to better reflect the relatedness of corresponding segments. The scoring scheme is not tied to any language-specific system of phonemic contrasts, but instead is derived from a universal set of phonetic features which is sufficiently general and flexible to express all possible speech sounds that exist in the world's languages. Similar schemes have been proposed before, but they all lacked the key concept of feature salience, which is necessary to properly balance the importance of various features. The algorithm was evaluated on an independently compiled set of cognates, and was found to outperform other algorithms reported in the literature.

In Chapter 5, I developed a method for identifying cognates directly from the vocabularies of related languages. The only requirement of the method is that the user provides two wordlists containing phonetically-transcribed lexemes with their meaning explained by short English glosses. The wordlists do not have to be synchronized, nor the lexemes forced into semantic slots. The phonetic transcription of orthographic data can be performed automatically for most languages. The overall similarity between vocabulary items is calculated as a linear combination of the phonetic and the semantic similarity. I showed that the phonetic measure proposed in Chapter 4 outdoes the orthographic measures commonly used in computational linguistics. The procedure for estimating semantic similarity, which employs an electronic lexical database and a part-of-speech tagger, is able to associate glosses that have little or no surface affinity. The experiments performed on actual vocabularies of four Algonquian languages demonstrate that the method not only identifies with good precision a large portion of cognates in the data, but also contributes new entries to the existing etymological dictionary.

In Chapter 6, I proposed an original approach to the determination of recurrent sound correspondences in bilingual wordlists based on the idea of relating recurrent correspondences between sounds to translational equivalences between words. Through induction of statistical models that are similar to those developed for statistical machine translation, the method is able to recover recurrent sound correspondences from bilingual wordlists that consist mostly of unrelated pairs. A cognate identification method that takes advantage of the discovered correspondences achieves higher accuracy than the previously reported algorithms. Finally, I showed how an algorithm designed for extracting noncompositional compounds from bitexts can be used to determine complex sound correspondences in wordlists. By applying the powerful expectation-maximization approach, a computer program can within minutes come close to replicating the results of prolonged linguistic investigations.

7.4 Conclusion

Although it is impossible to predict the impact that this thesis will have on research in computational linguistics, the responses that I have received so far indicate that the scope of applications extends well beyond language reconstruction. The phonetic alignment algorithm described in Chapter 4 has already been found sufficiently practical for implementation in research projects on declarative prosodic morphology [Girgsdies, 2000], and on aligning reference and hypothesized speech transcriptions [Fisher, 2000]. During informal discussions following my presentations at computational linguistics conferences, several experts commented positively on the relevance of my work on the identification of cognates to the issues arising in machine translation.

The experiments reported in this thesis provide a novel assessment of various algorithms and resources commonly employed in computational linguistics. Because genetic cognates arise by evolving from a single word in a proto-language, they are ideal for testing various lexical and semantic measures of similarity. An objective evaluation of such measures is difficult as evidenced by the fact that Brew and McKelvie [1996] devoted several days to judge the translational equivalence of several thousand word-pairs automatically extracted from a bilingual corpus. In contrast, the average precision of the lexical measures computed in Chapter 5 was instantly calculated by comparing it against the standard of previously established cognate sets. Other experiments that I performed confirm the generality of Melamed's algorithms for the induction of translational models and for the discovery of noncompositional compounds, and cast light on the quality of WordNet noun hierarchies.

The focus of this thesis has been the problem of automatic language reconstruction. I do not claim that the algorithms described here offer a comprehensive and fully satisfying solution to the problem. Indeed, I consider it unlikely that such a solution will be found soon. However, the results of the careful evaluations to which I submitted the proposed methods should leave no doubt that I have advanced the state of the art in the area of computational diachronic phonology. I hope that the set of publicly available programs that arise from this research will help comparative linguists to provide conclusive evidence for hitherto conjectural language groupings.

Appendix A

The Swadesh 200-word list

Many variants of the Swadesh 200-word list exist. The following version is used in the Comparative Indo-European Data Corpus [Dyen *et al.*, 1992]. Kessler [2001] employs another variation, which includes breast, claw, full, go, grease, horn, hot, human, knee, knife, moon, now, path, round, true; but not fat, fear, float, how, leg, live, person, right, road, rope, turn, walk, warm, when, where.

all
and
animal
ashes
at
back
bad
bark
because
belly
big
bird
to bite
black
blood
to blow
bone
to breathe
to burn
child
cloud
cold
to come
to count
to cut
day
to die
to dig
dirty
dog
to drink
dry
dull (knife)
dust
ear
earth
to eat
egg
eye
to fall
far
fat
father
to fear
feather
few
to fight
fire
fish
five
to float
to flow
flower
to fly
fog
foot
four
to freeze
fruit
to give
good
grass
green
guts
hair
hand
he
head
to hear
heart
heavy
here
to hit
hold
how
to hunt
husband
I
ice
if
in
to kill
know
lake
to laugh
leaf
left (hand)
leg
to lie (on side)
to live
liver
long
louse
man (male)
many
meat (flesh)
mother
mountain
mouth
name
narrow
near
neck
new
night
nose
not
old
one
other
person
to play
to pull
to push
to rain
red
right (correct)
right (hand)
river
road
root
rope
rotten
rub
salt
sand
to say
scratch
sea
to see
seed
to sew
sharp
short
to sing
to sit
skin
sky
to sleep
small
to smell
smoke
smooth
snake
snow
some
to spit
to split
to squeeze
to stab
to stand
star
stick
stone
straight
to suck
sun
to swell
to swim
tail
that
there
they
thick
thin
to think
this
thou
three
to throw
to tie
tongue
tooth
tree
to turn
two
to vomit
to walk
warm
to wash
water
we
wet
what
when
where
white
who
wide
wife
wind
wing
wipe
with
woman
woods
worm
you (plural)
year
yellow

Appendix B

A historical derivation program

This appendix describes a *Perl* program that simulates the phonological evolution of Polish, and its results. The program takes as input a Proto-Slavic etymon, transforms it by a series of sound changes, and produces a predicted modern Polish word. The descriptions of sound changes were extracted from textbooks on the historical phonology of Slavic. The changes are implemented as sequences of regular expressions substitutions. The program has no notion of phonological features, but uses sets of phonemes instead.

The program was applied to 706 Proto-Slavic etyma extracted from books on the historical phonology of Slavic. 80 (11.3%) of the Proto-Slavic etyma have no extant cognates in Polish because of the lexical replacement phenomenon. The results on the remaining 626 etyma are summarized in Table B.1. Most of the predictions match the actual Polish words exactly. In other cases, the phonemic edit distance betwen the prediction and the actual word is calculated. The percentage of exact matches could probably be increased by a more careful implemenation of the sound changes, but it is unlikely to reach 100%. Because of phenomena such as analogy and hypercorrection, sound changes are never perfectly regular.

Edit distance	Number of forms	Percent of total
0	454	72.5%
1	84	13.4%
2	63	10.1%
3	20	3.2%
4	5	0.8%
Total retained	626	100%

Table B.1: Tested proto-forms grouped by the accuracy of the output.

Some words that have undergone semantic shifts are given in Table B.2. Those words have survived, but their modern meaning has changed. Some of the words have retained their original meaning in other Slavic languages.

Proto-Slavic	Gloss	Generated form	Modern meaning
χrĭstŭ	‘cross’	chrzest	‘baptism’
iskati	‘to seek’	iskać	‘to groom’
jęʒa	‘disease’	jędza	‘witch’
napast’	‘temptation’	napaść	‘assault’
plŭtĭ	‘body’	płeć	‘sex’
tĭma	‘darkness, myriad’	ćma	‘moth’
volst’	‘power’	włość	‘estate, property’
znamenije	‘sign’	znamię	‘scar’

Table B.2: Examples of semantic shifts.

Table B.3 contains forms produced by applying the sequences of sound changes to the Proto-Slavic etyma that have been replaced by other words during the evolution of Polish. The words marked with a † are attested in Old Polish. What is interesting about the generated forms is that they conform to the phonotactic constraints of Polish. The words do sound Polish, but their meaning is totally opaque to the native Polish speakers.

Proto-Slavic	Gloss	Generated form	Modern meaning
čŕvĭ	'worm'	czerw†	?
děverĭ	'husband's brother'	dziewierz†	?
golgolŭ	'word'	głogoł	?
govorŭ	'noise'	gowor	?
jazva	'wound'	jazwa	?
klětĭ	'room'	kleć	?
kosnǫti	'to touch'	kosnąć	?
kovŭ	'ambush'	ków	?
krovŭ	'roof'	krów	?
kyslŭjĭ	'sour'	kisły†	?
lǫkavŭjĭ	'sly'	łąkawy	?
lišiti	'to deprive'	liszyć	?
l'ubodějica	'hussy'	lubodziejka	?
mĭzda	'wages'	mzda	?
napastĭ	'temptation'	napasć	?
nastavĭnikŭ	'supervisor'	nastawnik	?
nogŭtĭ	'claw'	nogieć	?
otročę	'child'	otroczę	?
pǫtĭ	'path'	pąć	?
poglumiti sę	'to ponder'	poglumić się	?
pospěšĭnikŭ	'helper'	pospiesznik	?
prěmuditi	'to delay'	przemudzić	?
rabŭ	'slave'	rab	?
sŭměrĭnŭjĭ	'humble'	śmierzny	?
skotŭ	'cattle'	skot†	?
stŭgna	'street'	stegna	?
taina	'secret'	tajna	?
umŭ	'mind'	um	?
velěti	'to order'	wieleć	?
vrěmę	'time'	wrzemię	?
χramŭ	'temple'	chrom	?
χytjĭnikŭ	'predator'	chycnik	?
žĭdati	'to wait'	żdać	?
žekti	'to burn'	żeć	?

Table B.3: Examples of the generated words that have no existing counterparts.

Appendix C

The alignment code

This appendix contains the C++ code for the phonetic alignment. The code, which has been constantly evolving since 1999, integrates several extensions of the basic dynamic programming algorithm described in Section 4.4. It constitutes the core of ALINE, and is also of prime importance in both COGIT and CORDI. By including the code, I hope to make it easier for other developers to incorporate advanced features into their alignment programs. Although I believe that it is essentially error-free, no warranty is implied.

```
/************************************************************************
The routine 'align' finds all alignments that are within a margin of the
optimal alignment, which is set by the parameter 'scoreMargin'.
However, if ONE_ONLY is defined, a single alignment is
constructed (which may not be optimal if scoreMargin < 1.0).

The following names correspond to four possible modes of comparison:
GLOBAL, SEMI_GLOBAL, HALF_LOCAL, and LOCAL.
Exactly one of them must be defined.
************************************************************************/

#include "external.h"

/*
The following external names are used:

const int MAXL;       // max length of a word
const int NUL;        // deletion filler
const int DUB;        // compression filler
const int LIM;        // local alignment delimiter
const int NoScore;    // very large negative integer

// A straightforward FIFO data structure.
class Stack
{
public:
    void clear();                    // remove all elements
    void push( int i1, int i2 = 0 ); // add element (i1) or (i1,i2) to the top
    void pop( short k = 1 );         // pop k elements from the top
    bool allowed();                  // false if insertion follows deletion
    void result( int );              // output alignment
};

// An interface between the alignment algorithm and the actual words.
class Sigma
{
public:
    short phlenA();                  // phonetic length of word A
    short phlenB();                  // phonetic length of word B
    int sub( short i, short j );     // cost of substituting A[i] with B[j]
    int expA( short i, short j );    // cost of expanding A[i] to B[j-1] B[j]
    int expB( short j , short i );   // cost of expanding B[j] to A[i-1] A[i]
    int skipA( short i );            // cost of deleting A[i]
    int skipB( short j );            // cost of deleting B[j]
};
*/

// global variables
int   S[MAXL][MAXL];            // the score matrix
float AcptScore; // minimal acceptable score
Stack Trace;                    // links between corresponding segments
Stack Out;                      // alignment constructed by dynamic programming
Stack Cost; // cost of individual operations
bool  FallThru;         // for ONE_ONLY mode
```

```
int similarity( Sigma *sig );
void alignment( Sigma *sig, short i, short j, int T );

// Wrapper function for the score matrix.
inline int Score( short i, short j )
{
    return ( ( i >= 0 ) && ( j >= 0 ) ? S[i][j] : NoScore );
}

// Finds all alignments within the scoreMargin.
void align( Sigma *sig, float scoreMargin = 1.0 )
{
    short lenA = sig->phlenA();
    short lenB = sig->phlenB();

    Cost.clear();
    Trace.clear();
    Out.clear();
    FallThru = false;

    int dpScore = similarity( sig );   // determine the maximum similarity
    AcptScore = dpScore * scoreMargin;

    for ( short i = 0; i <= lenA; i++ )
    {
        for ( short j = 0; j <= lenB; j++ )
        {
#if defined(GLOBAL) || defined(HALF_LOCAL)
            if ( i < lenA || j < lenB ) // corner start point only
                continue;
#endif
#if defined(SEMI_GLOBAL)
            if ( i < lenA && j < lenB ) // border start points only
                continue;
#endif
            if ( S[i][j] >= AcptScore )
            {
                // padding at the beginning
                for ( short j1 = lenB; j1 > j; j1-- )
                    Out.push( NUL, j1 );
                for ( short i1 = lenA; i1 > i; i1-- )
                    Out.push( i1, NUL );
                Out.push( LIM, LIM );           // delimits the alignment
                alignment( sig, i, j, 0 );      // recursion starts here
                Out.pop(lenA-i+lenB-j+1);
                if ( FallThru )
                    return;
            }
        }
    }
}
```

```
// Fills the score matrix S; returns the similarity between words A and B.
int similarity( Sigma *sig )
{
    short lenA = sig->phlenA();
    short lenB = sig->phlenB();
    int sgmax = 0; // not meaningful for global case
    int m1, m2, m3, m4, m5, lmax;

    S[0][0] = 0;

    for ( short i = 1; i <= lenA; i++ )
#if defined(GLOBAL)
        S[i][0] = S[i-1][0] + sig->skipA(i);
#else
        S[i][0] = 0;
#endif

    for ( short j = 1; j <= lenB; j++ )
#if defined(GLOBAL)
        S[0][j] = S[0][j-1] + sig->skipB(j);
#else
        S[0][j] = 0;
#endif

    for ( short i = 1; i <= lenA; i++ )
    {
        for ( short j = 1; j <= lenB; j++ )
        {
            m1 = Score(i-1,j) + sig->skipA(i);
            m2 = Score(i,j-1) + sig->skipB(j);
            m3 = Score(i-1,j-1) + sig->sub(i,j);
            m4 = Score(i-1,j-2) + sig->expA(i,j);
            m5 = Score(i-2,j-1) + sig->expB(j,i);
#if defined(LOCAL) || defined(HALF_LOCAL)
            lmax = max( m1, m2, m3, m4, m5, 0 );
#else
            lmax = max( m1, m2, m3, m4, m5 );
#endif
            S[i][j] = lmax;

            if ( lmax > sgmax )
#if defined(SEMI_GLOBAL)
                if ( i == lenA || j == lenB ) // border only for semi-global
#endif
                    sgmax = lmax;
        }
    }
#if defined(GLOBAL) || defined(HALF_LOCAL)
    dpScore = Score( lenA, lenB );
#endif
    return sgmax;
}
```

```
// Recursively retrieves alignments from the score matrix S;
// i and j indicate the current position within words A and B, respectively;
// T accumulates the total score of the current alignment.
void alignment( Sigma *sig, short i, short j, int T )
{
#ifdef ONE_ONLY
    if ( FallThru ) return;
#endif

    if ( i == 0 && j == 0 )
    {
record:
        assert( Score(i,j) == 0 );
        if ( Out.allowed() && !FallThru )
        {
            Out.push( LIM, LIM );                  // delimits the alignment
            // padding at the end
            for ( short i1 = i; i1 > 0; i1-- )
                Out.push( i1, NUL );
            for ( short j1 = j; j1 > 0; j1-- )
                Out.push( NUL, j1 );
            // output the alignment
#if defined(GLOBAL)
            Out.result( T + (i+j)*sig->skipA(0) );
#else
            Out.result( T );
#endif
            Out.pop(i+j+1);
#ifdef ONE_ONLY
            FallThru = true;
#endif
        }
    }
    else
    {
#if defined(LOCAL) || defined(HALF_LOCAL) || defined(SEMI_GLOBAL)
        if ( ( i == 0 ) || ( j == 0 ) ) goto record; // shortcut
#endif
        int subSc = sig->sub(i,j);
        if ( Score(i-1,j-1) + subSc + T >= AcptScore )
        {
            Cost.push( subSc );
            Out.push( i, j );
            Trace.push( i, j );
            alignment( sig, i-1, j-1, T + subSc );
            Trace.pop();
            Out.pop();
            Cost.pop();
        }

        int insSc = sig->skipB(j);
        if ( ( i == 0 ) || ( Score(i,j-1) + insSc + T >= AcptScore ) )
        {
            Cost.push( insSc );
```

```
            Out.push( NUL, j );
            alignment( sig, i, j-1, T + insSc );
            Out.pop();
            Cost.pop();
        }

        int expSc = sig->expA(i,j);
        if ( Score(i-1,j-2) + expSc + T >= AcptScore )
        {
            Cost.push( expSc );
            Cost.push( NUL );
            Out.push( i, j );
            Out.push( DUB, j-1 );
            Trace.push( i, j );
            Trace.push( i, j-1 );
            alignment( sig, i-1, j-2, T + expSc );
            Trace.pop(2);
            Out.pop(2);
            Cost.pop(2);
        }

        int delSc = sig->skipA(i);
        if ( ( j == 0 ) || ( Score(i-1,j) + delSc + T >= AcptScore ) )
        {
            Cost.push( delSc );
            Out.push( i, NUL );
            alignment( sig, i-1, j, T + delSc );
            Out.pop();
            Cost.pop();
        }

        int cmpSc = sig->expB(j,i);
        if ( Score(i-2,j-1) + cmpSc + T >= AcptScore )
        {
            Cost.push( cmpSc );
            Cost.push( NUL );
            Out.push( i, j );
            Out.push( i-1, DUB );
            Trace.push( i, j );
            Trace.push( i-1, j );
            alignment( sig, i-2, j-1, T + cmpSc );
            Trace.pop(2);
            Out.pop(2);
            Cost.pop(2);
        }
#if defined(LOCAL) || defined(HALF_LOCAL)
        if ( Score(i,j) == 0 ) goto record; // shortcut
#endif
    }
}
```

Appendix D

Covington's test set

This appendix contains the full set of alignments generated by ALINE on Covington's test set of 82 cognates. They are given in the right column. The left column reproduces the alignments produced by Covington's program. In the cases where Covington's program produces more than one alternative alignment, only the first one is given.

Spanish-French cognate pairs (I)

Covington's alignments	*ALINE's alignments*
yo/je	
y o	‖ y o ‖
ž ə	‖ ž ə ‖
tu/tu	
t u	‖ t u ‖
t ü	‖ t ü ‖
nosotros/nous	
n o s o t r o s	‖ n o ‖ sotros
n u – – – – – –	‖ n u ‖
quién/qui	
k y e n	‖ k ye ‖ n
k i – –	‖ k i ‖
qué/quoi	
k – e	‖ k e ‖
k w a	‖ k wa ‖
todos/tous	
t o d o s	‖ t o ‖ dos
t u – – –	‖ t u ‖
una/une	
u n a	‖ u n ‖ a
ü n –	‖ ü n ‖
dos/deux	
d o s	‖ d o ‖ s
d ö –	‖ d ö ‖
tres/troix	
t r – e s	‖ t r e ‖ s
t r w a –	‖ t r wa ‖
hombre/homme	
o m b r e	‖ o m ‖ bre
o m – – –	‖ o m ‖

Spanish-French cognate pairs (II)

Covington's alignments	*ALINE's alignments*
árbol/arbre	
a r b – o l	‖ a r b o l ‖
a r b r ə –	‖ a r b – r ‖ ə
pluma/plume	
p l u m a	‖ p l u m ‖ a
p l ü m –	‖ p l ü m ‖
cabeza/cap	
k a b e θ a	‖ k a b ‖ eθa
k a p – – –	‖ k a p ‖
boca/bouche	
b o k a	‖ b o k ‖ a
b u š –	‖ b u š ‖
pie/pied	
p y e	‖ p y e ‖
p y e	‖ p y e ‖
corazón/coeur	
k o r a θ o n	‖ k o r ‖ aθon
k ö r – – – –	‖ k ö r ‖
ver/voir	
b – e r	‖ b e r ‖
v w a r	‖ v wa r ‖
venir/venir	
b e n i r	‖ b e n i r ‖
v ə n i r	‖ v ə n i r ‖
decir/dire	
d e θ i r	‖ d e θ i r ‖
d – – i r	‖ d – – i r ‖
pobre/pauvre	
p o b r e	‖ p o b r e ‖
p o v r ə	‖ p o v r ə ‖

English-German cognate pairs (I)

Covington's alignments | *ALINE's alignments*

this/dieses

ð i – – s | || ð i s ||

d ī z e s | dī || z e s ||

that/das

ð æ t | || ð æ t ||

d a s | || d a s ||

what/was

w a t | || w a t ||

v a s | || v a s ||

not/nicht

n a – t | || n a – t ||

n i x t | || n i x t ||

long/lang

l o ŋ | || l o ŋ ||

l a ŋ | || l a ŋ ||

man/Mann

m æ n | || m æ n ||

m a n | || m a n ||

flesh/Fleisch

f l e – š | || f l e š ||

f l a y š | || f l ay š ||

blood/Blut

b l ə d | || b l ə d ||

b l ū t | || b l ū t ||

feather/Feder

f e ð ə r | || f e ð ə r ||

f ē d ə r | || f ē d ə r ||

hair/Haar

h æ r | || h æ r ||

h ā r | || h ā r ||

English-German cognate pairs (II)

<table>
<tr><th colspan="5">Covington's alignments</th><th colspan="7">ALINE's alignments</th></tr>
<tr><td colspan="5">ear/Ohr</td><td colspan="7"></td></tr>
<tr><td>i</td><td>r</td><td></td><td></td><td></td><td>||</td><td>i</td><td>r</td><td>||</td><td></td><td></td><td></td></tr>
<tr><td>ō</td><td>r</td><td></td><td></td><td></td><td>||</td><td>ō</td><td>r</td><td>||</td><td></td><td></td><td></td></tr>
<tr><td colspan="5">eye/Auge</td><td colspan="7"></td></tr>
<tr><td>a</td><td>–</td><td>–</td><td>y</td><td></td><td>||</td><td>a</td><td>y</td><td>||</td><td></td><td></td><td></td></tr>
<tr><td>a</td><td>w</td><td>g</td><td>ə</td><td></td><td>||</td><td>a</td><td>w</td><td>||</td><td>gə</td><td></td><td></td></tr>
<tr><td colspan="5">nose/Nase</td><td colspan="7"></td></tr>
<tr><td>n</td><td>o</td><td>w</td><td>z</td><td>–</td><td>||</td><td>n</td><td>ow</td><td>z</td><td>||</td><td></td><td></td></tr>
<tr><td>n</td><td>ā</td><td>–</td><td>z</td><td>ə</td><td>||</td><td>n</td><td>ā</td><td>z</td><td>||</td><td>ə</td><td></td></tr>
<tr><td colspan="5">mouth/Mund</td><td colspan="7"></td></tr>
<tr><td>m</td><td>a</td><td>w</td><td>–</td><td>θ</td><td>||</td><td>m</td><td>aw</td><td>–</td><td>θ</td><td>||</td><td></td></tr>
<tr><td>m</td><td>–</td><td>u</td><td>n</td><td>t</td><td>||</td><td>m</td><td>u</td><td>n</td><td>t</td><td>||</td><td></td></tr>
<tr><td colspan="5">tongue/Zunge</td><td colspan="7"></td></tr>
<tr><td>t</td><td>–</td><td>ə</td><td>ŋ</td><td>–</td><td></td><td>||</td><td>t</td><td>ə</td><td>ŋ</td><td>||</td><td></td></tr>
<tr><td>t</td><td>s</td><td>u</td><td>ŋ</td><td>ə</td><td>t</td><td>||</td><td>s</td><td>u</td><td>ŋ</td><td>||</td><td>ə</td></tr>
<tr><td colspan="5"></td><td>||</td><td>t</td><td>–</td><td>ə</td><td>ŋ</td><td>||</td><td></td></tr>
<tr><td colspan="5"></td><td>||</td><td>t</td><td>s</td><td>u</td><td>ŋ</td><td>||</td><td>ə</td></tr>
<tr><td colspan="5">foot/Fuß</td><td colspan="7"></td></tr>
<tr><td>f</td><td>u</td><td>t</td><td></td><td></td><td>||</td><td>f</td><td>u</td><td>t</td><td>||</td><td></td><td></td></tr>
<tr><td>f</td><td>ū</td><td>s</td><td></td><td></td><td>||</td><td>f</td><td>ū</td><td>s</td><td>||</td><td></td><td></td></tr>
<tr><td colspan="5">knee/Knie</td><td colspan="7"></td></tr>
<tr><td>-</td><td>n</td><td>i</td><td>y</td><td></td><td></td><td>||</td><td>n</td><td>iy</td><td>||</td><td></td><td></td></tr>
<tr><td>k</td><td>n</td><td>ī</td><td>–</td><td></td><td>k</td><td>||</td><td>n</td><td>ī</td><td>||</td><td></td><td></td></tr>
<tr><td colspan="5">hand/Hand</td><td colspan="7"></td></tr>
<tr><td>h</td><td>æ</td><td>n</td><td>d</td><td></td><td>||</td><td>h</td><td>æ</td><td>n</td><td>d</td><td>||</td><td></td></tr>
<tr><td>h</td><td>a</td><td>n</td><td>t</td><td></td><td>||</td><td>h</td><td>a</td><td>n</td><td>t</td><td>||</td><td></td></tr>
<tr><td colspan="5">heart/Herz</td><td colspan="7"></td></tr>
<tr><td>h</td><td>a</td><td>r</td><td>t</td><td>–</td><td>||</td><td>h</td><td>a</td><td>r</td><td>t</td><td>||</td><td></td></tr>
<tr><td>h</td><td>e</td><td>r</td><td>t</td><td>s</td><td>||</td><td>h</td><td>e</td><td>r</td><td>t</td><td>||</td><td>s</td></tr>
<tr><td colspan="5">liver/Leber</td><td colspan="7"></td></tr>
<tr><td>l</td><td>i</td><td>v</td><td>ə</td><td>r</td><td>||</td><td>l</td><td>i</td><td>v</td><td>ə</td><td>r</td><td>||</td></tr>
<tr><td>l</td><td>ē</td><td>b</td><td>ə</td><td>r</td><td>||</td><td>l</td><td>ē</td><td>b</td><td>ə</td><td>r</td><td>||</td></tr>
</table>

English-Latin cognate pairs (I)

Covington's alignments							*ALINE's alignments*							
and/ante														
æ	n	d	–				||	æ	n	d	||			
a	n	t	e				||	a	n	t	||	e		
at/ad														
æ	t						||	æ	t	||				
a	d						||	a	d	||				
blow/flāre														
b	l	–	–	o	w		||	b	l	o	||	w		
f	l	ā	r	e	–		||	f	l	ā	||	re		
ear/auris														
i	–	r	–	–			||	i	r	||				
a	w	r	i	s			||	aw	r	||	is			
eat/edere														
i	y	t	–	–	–		||	iy	t	||				
e	–	d	e	r	e		||	e	d	||	ere			
fish/piscis														
-	–	–	f	i	š		||	f	i	š	||			
p	i	s	k	i	s		||	p	i	s	||	kis		
flow/fluere														
f	l	o	w	–	–	–	||	f	l	ow	||			
f	l	–	u	e	r	e	||	f	l	u	||	ere		
star/stēlla														
s	t	a	r	–	–		||	s	t	a	r	||		
s	t	ē	l	l	a		||	s	t	ē	l	||	la	
full/plēnus														
-	–	–	f	u	l		||	f	u	l	||			
p	l	ē	n	u	s		||	p	–	l	||	ēnus		
grass/grāmen														
g	r	–	–	æ	s		||	g	r	æ	||	s		
g	r	ā	m	e	n		||	g	r	ā	||	men		
heart/cordis														
h	a	r	–	–	t		||	h	a	r	t	||		
k	o	r	d	i	s		||	k	o	r	d	||	is	
horn/cornū														
h	o	r	n	–			||	h	o	r	n	||		
k	o	r	n	ū			||	k	o	r	n	||	ū	
I/ego														
-	–	a	y				||	ay	||					
e	g	o	–				||	e	||	go				

English-Latin cognate pairs (II)

Covington's alignments	*ALINE's alignments*
knee/genū	
- – n i y	‖ n i ‖ y
g e n ū –	ge ‖ n ū ‖
mother/māter	
m ə ð ə r	‖ m ə ð ə r ‖
m ā t e r	‖ m ā t e r ‖
mountain/mōns	
m a w n t ə n	‖ m aw n t ‖ ə n
m ō – n – – s	‖ m ō n s ‖
name/nōmen	
n e y m – –	‖ n ey m ‖
n ō – m e n	‖ n ō m ‖ en
new/novus	
n y u w – –	‖ n yu w ‖
n – o w u s	‖ n o w ‖ us
one/ūnus	
w ə n – –	‖ wə n ‖
- ū n u s	‖ ū n ‖ us
round/rotundus	
r a – w n d – –	‖ r a – w n d ‖
r o t u n d u s	‖ r o t u n d ‖ us
sew/suere	
s o w – – –	‖ s ow ‖
s – u e r e	‖ s u ‖ ere
sit/sēdere	
s i t – – –	‖ s i t ‖
s ē d e r e	‖ s ē d ‖ ere
three/trēs	
θ r i y	‖ θ r iy ‖
t r ē s	‖ t r ē ‖ s
tooth/dentis	
- – – t u w θ	‖ t uw θ ‖
d e n t i – s	den ‖ t i s ‖
thin/tenuis	
θ i n – – –	‖ θ i n ‖
t e n u i s	‖ t e n ‖ uis

Fox-Menomini cognate pairs

Covington's alignments	*ALINE's alignments*
kiinwaawa/kenuaq	
k ī n w ā w a –	‖ k ī n w ā ‖ wa
k e n – – u a ʔ	‖ k e n u a ‖ ʔ
niina/nenah	
n ī n a –	‖ n ī n a ‖
n e n a h	‖ n e n a ‖ h
naapeewa/naapɛɛw	
n ā p ē w a	‖ n ā p ē w ‖ a
n ā p ɛ̄ w –	‖ n ā p ɛ̄ w ‖
waapimini/waapemen	
w ā p i m i n i	‖ w ā p i m i n ‖ i
w ā p e m e n –	‖ w ā p e m e n ‖
nameesa/namɛɛqs	
n a m ē – s a	‖ n a m ē – s ‖ a
n a m ɛ̄ ʔ s –	‖ n a m ɛ̄ ʔ s ‖
okimaawa/okeemaaw	
o k i m ā w a	‖ o k i m ā w ‖ a
o k ē m ā w –	‖ o k ē m ā w ‖
šišiipa/seeqsep	
š ī – š ī p a	‖ š ī - š ī p ‖ a
s ē ʔ s e p –	‖ s ē ʔ s e p ‖
ahkohkwa/ahkɛɛh	
a h k o h k w a	‖ a h k o h ‖ kwa
a h k ɛ̄ h – – –	‖ a h k ɛ̄ h ‖
pemaatesiweni/pemaateswen	
p e m ā t e s i w e n i	‖ p e m ā t e s i w e n ‖ i
p e m ā t e s e w e n –	‖ p e m ā t e s e w e n ‖
asenya/aqsɛn	
a – s e n y a	‖ a – s e n ‖ ya
a ʔ s ɛ n – –	‖ a ʔ s ɛ n ‖

Cognate pairs from other languages

Covington's alignments	*ALINE's alignments*
didōmi/dō	
d i d ō m i	di ‖ d ō ‖ mi
- – d ō – –	‖ d ō ‖
thugatēr/Tochter	
t^h u g a t ē r	‖ t^h u g a t ē r ‖
t o x – t ə r	‖ t o x – t ə r ‖
daughter/thugatēr	
- – d o t ə r	‖ d o t ə r ‖
t^h u g a t ē r	t^hu ‖ g a t ē r ‖
ager/ajras	
a – g e r	‖ a g e r ‖
a ǰ r a s	‖ a ǰ – r ‖ as
bharāmi/pherō	
b^h a r ā m i	‖ b^h a r ā ‖ mi
p^h e r – – o	‖ p^h e r o ‖
centum/hekaton	
- – k e n t u m	‖ k e n t u m ‖
h e k a – t o n	he ‖ k a – t o n ‖
centum/satəm	
k e n t u m	‖ k e n t u m ‖
s a – t ə m	‖ s a – t ə m ‖

Appendix E

A list of English–Latin cognates

This list of 112 established English–Latin cognate pairs has been taken from linguistic literature. It is not meant to be complete, and it may contain a few errors. English words were transcribed into phonetic notation with the aid of the publicly available Carnegie Mellon Pronouncing Dictionary, version 0.6.

ēkər	agr	‘acre’	dū	feki	‘do’
ægnēl	angust	‘agnail’	dɔr	fore	‘door’
ɔldər	aln	‘alder’	ɪr	aur	‘ear’
æm	sum	‘am’	īt	ed	‘eat’
æš	orn	‘ash’	ēt	okto	‘eight’
æksəl	aks	‘axle’	ɛlbō	uln	‘elbow’
bī	fuit	‘be’	yū	ow	‘ewe’
bɛr	fer	‘bear’	ay	okul	‘eye’
bɪrd	barb	‘beard’	fɛrō	pork	‘farrow’
bīč	fag	‘beech’	faðər	patr	‘father’
bərč	fraksin	‘birch’	fɛðər	penn	‘feather’
blēd	foli	‘blade’	fīd	pask	‘feed’
brəðər	fratr	‘brother’	fīld	plan	‘field’
kōm	gemm	‘comb’	fɪlm	pell	‘film’
kɔrn	gran	‘corn’	fayv	kwinkwe	‘five’
kaw	bowe	‘cow’	flī	pulik	‘flea’

flō	plu	‘flow’
fōm	spum	‘foam’
fut	pede	‘foot’
fɔr	kwart	‘four’
ful	plen	‘full’
gardən	hort	‘garden’
gūs	ansr	‘goose’
gɛst	host	‘guest’
hart	korde	‘heart’
hɔrn	kornu	‘horn’
hawnd	kan	‘hound’
həndrəd	kent	‘hundred’
ay	ego	‘I’
ɪz	est	‘is’
kɪn	gen	‘kin’
nī	genu	‘knee’
nō	gnosk	‘know’
lɪk	ling	‘lick’
lay	lekt	‘lie’
layt	lew	‘light’ (adj)
layt	luk	‘light’ (noun)
lɪsən	klu	‘listen’
lɔŋ	long	‘long’
ləv	lube	‘love’
mīl	mol	‘meal’
mɛlt	moll	‘melt’
mɪr	mare	‘mere’
mɪkəl	magn	‘mickle’
mɪdəl	medi	‘middle’
maynd	mente	‘mind’
mənθ	mens	‘month’
məðər	matr	‘mother’
maws	mur	‘mouse’
mərdər	morte	‘murder’
nēl	ungw	‘nail’
nēkəd	nud	‘naked’
nēm	nomine	‘name’
nɛfyū	nepot	‘nephew’
nɛst	nid	‘nest’
nū	now	‘new’
nayt	nokte	‘night’
nayn	nowem	‘nine’
nōz	nas	‘nose’
nat	non	‘not’
ra	kruor	‘raw’
rɛd	rube	‘red’
rūt	radike	‘root’
sɔlt	sal	‘salt’
sīd	semen	‘seed’
sɛvən	septem	‘seven’
sō	su	‘sew’
šɔrt	kurt	‘short’
sɪstər	soror	‘sister’
sɪt	sedet	‘sit’
sɪks	seks	‘six’
snō	niwem	‘snow’
spɪt	spu	‘spit’
stænd	stare	‘stand’
star	stell	‘star’
sək	sug	‘suck’
sən	sol	‘sun’
swīt	suaw	‘sweet’
tɛn	dekem	‘ten’
θɪn	tenu	‘thin’
θrī	tres	‘three’
θrəš	turd	‘thrush’
təŋ	lingw	‘tongue’
tūθ	dente	‘tooth’
trī	dur	‘tree’
tū	duo	‘two’
wɔrm	form	‘warm’
wɔtər	und	‘water’
hū	kwis	‘who’
wɪdō	widu	‘widow’
waynd	went	‘wind’
wulf	lup	‘wolf’
wul	lan	‘wool’
rɪŋ	wert	‘wring’
yōk	iug	‘yoke’
yəŋ	yuwenk	‘young’

Bibliography

[Adamson and Boreham, 1974] George W. Adamson and Jillian Boreham. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. *Information Storage and Retrieval*, 10:253–260, 1974.

[Al-Onaizan *et al.*, 1999] Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, D. Melamed, F. Och, D. Purdy, N. Smith, and D. Yarowsky. Statistical machine translation. Technical report, Johns Hopkins University, 1999.

[Allen, 1995] James Allen. *Natural Language Understanding.* The Benjamin Cummings Publishing Company, second edition, 1995.

[Baxter and Ramer, 2000] William H. Baxter and Alexis Manaster Ramer. Beyond lumping and splitting: probabilistic issues in historical linguistics. In Colin Renfrew, April McMahon, and Larry Trask, editors, *Time Depth in Historical Linguistics*, pages 167–188. The McDonald Institute for Archeological Research, 2000.

[Bloomfield, 1925] Leonard Bloomfield. On the sound-system of central Algonquian. *Language*, 1:130–156, 1925.

[Bloomfield, 1946] Leonard Bloomfield. Algonquian. In Harry Hoijer et al., editor, *Linguistic Structures of Native America*, volume 6 of *Viking Fund Publications in Anthropology*, pages 85–129. New York: Viking, 1946.

[Brew and McKelvie, 1996] Chris Brew and David McKelvie. Word-pair extraction for lexicography. In K. Oflazer and H. Somers, editors, *Proceedings of the 2nd International Conference on New Methods in Language Processing*, pages 45–55, Ankara, Bilkent University, 1996.

[Brill, 1995] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–566, 1995.

[Brown *et al.*, 1990a] P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[Brown *et al.*, 1990b] P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1990.

[Buck, 1949] Carl Darling Buck. *A dictionary of selected synonyms in the principal Indo-European languages: a contribution to the history of ideas.* Universtity of Chicago Press, 1949.

[Budanitsky, 1999] Alexander Budanitsky. Lexical semantic relatedness and its application in natural language processing. Technical Report CSRG-390, University of Toronto, 1999. Available at ftp.cs.toronto.edu/csrg-technical-reports.

[Burton-Hunter, 1976] Sarah K. Burton-Hunter. Romance etymology: a computerized model. *Computers and the Humanities*, 10:217–220, 1976.

[Chomsky and Halle, 1968] Noam Chomsky and Morris Halle. *The Sound Pattern of English.* New York: Harper & Row, 1968.

[Church, 1993] Kenneth W. Church. Char_align: A program for aligning parallel texts at the character level. In *Proceedings of ACL-93: 31st Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Columbus, Ohio, 1993.

[Connolly, 1997] John H. Connolly. Quantifying target-realization differences. *Clinical Linguistics & Phonetics*, 11:267–298, 1997.

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

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

[Dayhoff *et al.*, 1983] M. O. Dayhoff, W. C. Baker, and L.T. Hunt. Establishing homologies in protein sequences. *Methods in Enzymology*, 91:524–545, 1983.

[Dunning, 1993] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[Durbin *et al.*, 1998] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis*. Cambridge University Press, 1998.

[Dyen *et al.*, 1992] Isidore Dyen, Joseph B. Kruskal, and Paul Black. An Indoeuropean classification: A lexicostatistical experiment. *Transactions of the American Philosophical Society*, 82(5), 1992. Word lists available at http://www.ldc.upenn.edu/ldc/service/comp-ie.

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

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

[Fellbaum, 1998] Christiane Fellbaum, editor. *WordNet: an electronic lexical database*. The MIT Press, Cambridge, MA, 1998.

[Fisher, 2000] William Fisher. Personal communication, 2000.

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

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

[Girgsdies, 2000] Stefan Girgsdies. Personal communication, 2000.

[Gotoh, 1982] Osamu Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708, 1982.

[Greenberg, 1987] Joseph H. Greenberg. *Language in the Americas.* Stanford University Press, 1987.

[Greenberg, 1993] Joseph H. Greenberg. Observations concerning Ringe's *Calculating the factor of chance in language comparison. Proceedings of the American Philosophical Society*, 137:79–89, 1993.

[Guy, 1994] Jacques B. M. Guy. An algorithm for identifying cognates in bilingual wordlists and its applicability to machine translation. *Journal of Quantitative Linguistics*, 1(1):35–42, 1994. MS-DOS executable available at http://garbo.uwasa.fi.

[Hall, 1976] Robert A. Hall. *Proto-Romance phonology.* New York: Elsevier, 1976.

[Hartman, 1981] Steven Lee Hartman. A universal alphabet for experiments in comparative phonology. *Computers and the Humanities*, 15:75–82, 1981.

[Hewson, 1974] John Hewson. Comparative reconstruction on the computer. In *Proceedings of the 1st International Conference on Historical Linguistics*, pages 191–197, 1974.

[Hewson, 1989] John Hewson. Computer-aided research in comparative and historical linguistics. In I. Batori, W. Lenders, and W. Putschke, editors, *Computational linguistics: an international handbook on computer oriented language research and applications*, pages 576–580. Berlin: W. de Gruyter, 1989.

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

[Hewson, 1999] John Hewson. Vocabularies of Fox, Cree, Menomini, and Ojibwa, 1999. Computer file.

[Johnson, 1985] Mark Johnson. Computer aids for comparative dictionaries. *Linguistics*, 23(2):285–302, 1985.

[Kay, 1964] Martin Kay. The logic of cognate recognition in historical linguistics. Memorandum RM-4224-PR, The RAND Corporation, Santa Monica, September 1964.

[Kessler, 1995] Brett Kessler. Computational dialectology in Irish Gaelic. In *Proceedings of EACL-95: 6th Conference of the European Chapter of the Association for Computational Linguistics*, pages 60–67, 1995.

[Kessler, 2001] Brett Kessler. *The Significance of Word Lists*. Stanford: CSLI Publications, 2001. Word lists available at http://spell.psychology.wayne.edu/∼bkessler.

[Knight and Graehl, 1998] Kevin Knight and Jonathan Graehl. Machine transliteration. *Computational Linguistics*, 24(4):599–612, 1998.

[Koehn and Knight, 2001] Philipp Koehn and Kevin Knight. Knowledge sources for word-level translation models. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 27–35, 2001.

[Kondrak, 2000] Grzegorz Kondrak. A new algorithm for the alignment of phonetic sequences. In *Proceedings of NAACL 2000: 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 288–295, 2000.

[Kondrak, 2001a] Grzegorz Kondrak. Identifying cognates by phonetic and semantic similarity. In *Proceedings of NAACL 2001: 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 103–110, 2001.

[Kondrak, 2001b] Grzegorz Kondrak. Review of Brett Kessler's *The significance of world lists*. *Computational Linguistics*, 27:588–591, 2001.

[Kondrak, 2002] Grzegorz Kondrak. Determining recurrent sound correspondences by inducing translation models. In *Proceedings of COLING 2002: 19th International Conference on Computational Linguistics*, 2002. To appear.

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

[Ladefoged, 1995] Peter Ladefoged. *A Course in Phonetics.* New York: Harcourt Brace Jovanovich, 1995.

[Leacock and Chodorow, 1998] Claudia Leacock and Martin Chodorow. Combining local context and WordNet similarity for word sense identification. In Christiane Fellbaum, editor, *WordNet: an electronic lexical database*, pages 265–283. The MIT Press, Cambridge, MA, 1998.

[Lin, 1998] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304, 1998.

[Lowe and Mazaudon, 1994] John B. Lowe and Martine Mazaudon. The reconstruction engine: a computer implementation of the comparative method. *Computational Linguistics*, 20:381–417, 1994.

[Lowrance and Wagner, 1975] Roy Lowrance and Robert A. Wagner. An extension of the string-to-string correction problem. *Journal of the Association for Computing Machinery*, 22:177–183, 1975.

[Mann and Yarowsky, 2001] Gideon S. Mann and David Yarowsky. Multipath translation lexicon induction via bridge languages. In *Proceedings of NAACL 2001: 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 151–158, 2001.

[McEnery and Oakes, 1996] Tony McEnery and Michael Oakes. Sentence and word alignment in the CRATER Project. In J. Thomas and M. Short, editors, *Using Corpora for Language Research*, pages 211–231. Longman, 1996.

[Melamed, 1997] I. Dan Melamed. Automatic discovery of non-compositional compounds in parallel data. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 97–108, 1997.

[Melamed, 1999] I. Dan Melamed. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1):107–130, 1999.

[Melamed, 2000] I. Dan Melamed. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, 2000.

[Myers, 1995] Eugene W. Myers. Seeing conserved signals. In Eric S. Lander and Michael S. Waterman, editors, *Calculating the Secrets of Life*, pages 56–89. Washington, D.C.: National Academy Press, 1995.

[Nerbonne and Heeringa, 1997] John Nerbonne and Wilbert Heeringa. Measuring dialect distance phonetically. In *Proceedings of SIGPHON-97: 3rd Meeting of the ACL Special Interest Group in Computational Phonology*, 1997. Available at http://www.cogsci.ed.ac.uk/sigphon.

[Nerbonne *et al.*, 1996] John Nerbonne, Wilbert Heeringa, Erik van den Hout, Peter van der Kooi, Simone Otten, and Willem van de Vis. Phonetic distance between Dutch dialects. In G. Durieux, W. Daelemans, and S. Gillis, editors, *CLIN VI: Proceedings of the Sixth CLIN Meeting*, pages 185–202, Antwerp, Centre for Dutch Language and Speech (UIA), 1996.

[Nerbonne *et al.*, 1999] John Nerbonne, Wilbert Heeringa, and Peter Kleiweg. Edit distance and dialect proximity. In David Sankoff and Joseph B. Kruskal, editors, *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*, pages v–xv. Stanford: CSLI Publications, 1999.

[Oakes, 2000] Michael P. Oakes. Computer estimation of vocabulary in protolanguage from word lists in four daughter languages. *Journal of Quantitative Linguistics*, 7(3):233–243, 2000.

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

[Oommen, 1995] B. John Oommen. String alignment with substitution, insertion, deletion, squashing, and expansion operations. *Information Sciences*, 83:89–107, 1995.

[Oswalt, 1998] Robert L. Oswalt. A probabilistic evaluation of North Eurasiatic Nostratic. In Joseph C. Salmons and Brian D. Joseph, editors, *Nostratic: sifting the evidence*, pages 199–216. Amsterdam: John Benjamins, 1998.

[Raman *et al.*, 1997] Anand Raman, John Newman, and Jon Patrick. A complexity measure for diachronic chinese phonology. In *Proceedings of SIGPHON-97: Third Meeting of the ACL Special Interest Group in Computational Phonology*, 1997.

[Remmel, 1979] Mart Remmel. Computer techniques in Balto-Finnic historical phonetics. Technical Report Preprint KKI–11, Academy of Sciences of the Estonian S.S.R., 1979.

[Remmel, 1980] Mart Remmel. Computers in the historical phonetics and phonology of Balto-Finnic languages: problems and perspectives. Technical Report Preprint KKI–14, Academy of Sciences of the Estonian S.S.R., 1980.

[Rennie, 1999] Jason Rennie. Wordnet::QueryData Perl module, 1999. Available at http://www.ai.mit.edu/∼jrennie.

[Resnik, 1999] Philip Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[Ringe, 1992] Don Ringe. On calculating the factor of chance in language comparison. *Transactions of the American Philosophical Society*, 82(1), 1992.

[Ringe, 1998] Don Ringe. Probabilistic evidence for Indo-Uralic. In Joseph C. Salmons and Brian D. Joseph, editors, *Nostratic: sifting the evidence*, pages 153–197. Amsterdam: John Benjamins, 1998.

[Simard *et al.*, 1992] Michel Simard, George F. Foster, and Pierre Isabelle. Using cognates to align sentences in bilingual corpora. In *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 67–81, Montreal, Canada, 1992.

[Smith and Waterman, 1981] T. F. Smith and Michael S. Waterman. Identification of common molecular sequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[Smith, 1969] Raoul N. Smith. A computer simulation of phonological change. *ITL: Tijdschrift voor Toegepaste Linguistiek*, 1(5):82–91, 1969.

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

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

[Somers, 2000] Harold L. Somers. Personal communication, 2000.

[Swadesh, 1952] Morris Swadesh. Lexico-statistical dating of prehistoric ethnic contacts. *Proceedings of the American Philosophical Society*, 96:452–463, 1952.

[Tiedemann, 1999] Jörg Tiedemann. Automatic construction of weighted string similarity measures. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, College Park, Maryland, 1999.

[Trask, 1996] R. L. Trask. *Historical Linguistics*. London: Arnold, 1996.

[Vossen, 1998] Piek Vossen, editor. *EuroWordNet: a Multilingual Database with Lexical Semantic Networks*. Kluwer Academic, Dordrecht, 1998.

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