2010-11-22

PyAline: Automatically Growing Language Family Trees Using the ALINE Distance

Paul A. Huff
Brigham Young University - Provo

PyAline: Automatically Growing Language Family Trees

Using The ALINE Distance

Paul Huff

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of the Arts

Deryle Lonsdale, Chair
David Eddington
Dirk Elzinga

Department of Linguistics

Brigham Young University

December 2010

ABSTRACT

PyAline: Automatically Growing Language Family Trees

Using The ALINE Distance

Paul Huff

Department of Linguistics

Master of the Arts

Several methods for determining a numerical distance between languages have been proposed in the literature. In this thesis I implement one of them, the ALINE distance, and develop a methodology for comparing its results with other language distance metrics. I then compare it with a leading distance metric, the LDND distance, proposed by the ASJP project.

Keywords: language phylogenies automatic language classification

ACKNOWLEDGMENTS

I would like to thank my advisor, Deryle Lonsdale. Dr. Lonsdale's advice and patience with me as I have slowly completed my degree while working full-time have allowed him to gently nudge me in the right direction to stay on course through several hurdles in this process. I would also like to thank Dr. Eddington and Dr. Elzinga for their patience and willingness to deal with a part-time grad student as committee members.

Søren Wichmann was instrumental in allowing me access to the ASJP project and then letting me bounce ideas off him for how to compare PyAline's results with those given by the ASJP project. He also has been wonderful at helping me test and publish the results of the side projects that my thesis generated.

Amazon.com provided me with two cluster computing grants that made it possible for me to use their EC2 computing system. Without this, I would never have been able to actually process a language group with more than 150 languages, making most of my larger measurements impossible.

My employers throughout my Masters program, first Sourceforge and then Ning, have been more than patient and willing to let me take off the time necessary to work on this degree and thesis.

Lastly, and most importantly, I would like to thank my family. Melissa, my wife, (who decided that she would go to grad school along with me and then finished first!), has been the most patient and helpful companion one could hope for in this process, reading drafts and slogging through algorithms and helping me to see the light at the end of the tunnel. Caleb has provided frequent needed joyful distractions and has scribbled his own thesis chapters alongside me, and Cecily has cuddled her way into our hearts, being born just days before I defended.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The use of statistical and numerical methods to analyze language change and relatedness has experienced a resurgence in interest over the last decade, despite having been relatively understudied over the last 50 years. Among other things two extra-linguistic advances have helped feed this resurgence. The first is the development of new statistical methods pioneered by mathematicians and biologists for understanding tree-like structures. The second is the continuing increase in the speed and power of computers available to the average linguist. These two advances have helped make it easy to test statistical hypotheses regarding language. Consequently, lexicostatistics as a linguistic subdiscipline has suddenly become a much more compelling field of research to a growing number of linguists.

Over the last twenty to thirty years statistics have been put to great use for a variety of linguistic tasks. However, statistics have only recently been used in earnest to study what might be called the original questions that founded the field of linguistics itself: How did all known languages come to be as they are and how are they related to one another? For roughly the first two hundred years of the field's life, linguists were nearly universally interested in such questions. For a brief period in the 1950s and 1960s linguists began to look at these questions of language origin and relatedness using statistical methods, but for much of the last fifty years historical linguists have eschewed their use. The recent return of statistical methods to the historical analysis of language is therefore as much about the field of historical linguistics itself as it is a potential new way to shed more light

on the subject.

Historical linguists have traditionally used the historical-comparative method as the principal means to elucidate the relationships between languages. This method requires historical linguists to look at a group of languages' similarities and differences, construct a model of how their unattested ancestors must have evolved into their current synchronic state, and then group the languages together into language families based on shared innovations from their ancestor languages [McMahon and McMahon, 2005]. The method was developed (and used quite successfully) to discover the relationships among the various Indo-European languages, and has been extended successfully to various other language families throughout the world.

As statistical and mathematical methods became more powerful during the first half of the twentieth century, they became an appealing tool to linguists. Perhaps the most infamous mathematical method used for discovering the relationships between languages is glottochronology, championed in the 1950s by Morris Swadesh and others. Swadesh sought to investigate the historical relationships between languages by creating lists for certain culturally universal lexical items. Glottochronology relies on the fact that so-called basic vocabulary items change less frequently than other vocabulary items; by finding basic, universal lexical items, the rate of change across various languages ought to be relatively constant. By looking at the number of items that were similar or cognate between the lists of basic vocabulary between two related languages and making the assumption of a constant rate of change of any given language, practitioners of glottochronology sought to establish with mathematical rigor a model for showing when languages descended from one another. The principal glottochronological equation is as follows (following McMahon and McMahon):

$$t = \frac{\log c}{2 \log r}$$

where t = time depth in millennia, c = percentage of cognates and r = the glottochronological constant. Using this formula, Swadesh and others attempted to discern the exact split dates in

millennia between languages and their ancestors [McMahon and McMahon, 2005].

Glottochronology's methods were heavily criticized and eventually nearly universally rejected on a variety of grounds. Critics attacked the methodology of list construction, saying that there are very few basic lexical items that are in reality universally present across all cultures. They also attacked the rigor of the list construction process, since in some languages there are lexical-semantic pairs for which more than one word might fit. Since the accuracy of the lists themselves is suspect, the portion of the glottochronological equation which represents the percentage of cognates is suspect as well, as it is derived from the lists. Finally glottochronology's assumptions of a constant rate of change was most roundly criticized since using the formula leads to time depths which vary widely from known time depths. For some language pairs, like Tok Pisin and English, the equation estimated the divergence as being much older than it actually was. For other language pairs, such as Old Norse and Icelandic and Old Armenian and Modern Armenian, the glottochronology equation predicts language deviation times which are vastly later than they actually were. These inconsistencies highlighted the difficulty of trying to model language deviation times with a single constant rate of change. Glottochronology's mathematical problems contributed to a long disparagement of the use of numerical methods in historical linguistics [McMahon and McMahon, 2005].

Another relatively well-known attempt at finding other ways of modelling language relatedness was proposed and refined by Joseph Greenberg. "Multilateral comparison" was the name he gave to the set of techniques and principles he developed in the 1950s and 1960s for creating genetic family trees of languages. Essentially he gathered lists of form-meaning pairs for sets of languages similar to Swadesh's. Then he used a complex set of mathematical formulas and factors for judging similarities across large numbers of languages at a time, rather than just single languages. At least ten different phenomena were considered as decisive, and were ranked in order. If a set of languages had a similar form-meaning pair, then the languages could be grouped together according to the mathematical weighting of the similarity between the forms. More similarities between

more pairs meant that there was much more likely to be a genetic grouping than if a group of languages simply shared one cognate between them [Croft, 2005].

Although much of what Greenberg proposed seemed sound initially, his methods were also fairly universally criticized by the mainstream historical linguistic community. The technology simply wasn't available at the time to allow him to perform his (quite rigorous) statistical calculations transparently, making the underlying processes seem somewhat opaque to his critics. Indeed, many criticized the sloppiness and seeming arbitrariness of his assumptions of similarity [Croft, 2005]. They likewise criticized his use of error-ridden data, his failure to use cognates that were known to be genuine but dissimilar, the large number of languages he investigated at a time, and several other methodological choices [Croft, 2005].

The general and quite public rejection of both Greenberg's multilateral comparison and Swadesh's glottochronology by the historical linguistics community guaranteed that no further mathematical models of language relatedness were seriously entertained for quite some time. Because of the rather dramatic and public failures of these kinds of mathematical models to gain traction among historical linguists, one might ask why anyone would bother returning to mathematical methods for learning about linguistics. It is certainly true that the historical-comparative method's long-standing valuable contributions to the body of historical knowledge of languages and linguistics are indisputable.

However, despite the historical-comparative method's successes, Calvert Watkins, the noted Indo-Europeanist, made clear that the method is not an end in and of itself, but instead is a means to the end goal of understanding the history of how the world's languages came to be as they are:

> The reconstruction of Indo-European [via the historical-comparative method], the establishment, that is, of the grammar of that language to the best of our ability, is not our fundamental object, as it would be if we were writing a descriptive grammar of a known language. Rather, our ultimate aim is to write the linguistic history of known

> languages. We are seeking historical explanation for the grammar of languages accessible to us by observation or from written texts. Reconstruction is only a tool, a means to the end of understanding linguistic history.
>
> Even if we were, by some miracle, handed a complete grammar of Common Indo-European as spoken somewhere in, say, 4000 B.C. (the date is meaningless), the work of the Indo-Europeanist would scarcely be done. In fact, it would be barely begun. For his task would be, then as before, to relate the facts vouchsafed him to the facts of attested languages: to construct hypotheses, and to demonstrate precisely how it is possible, within a linguistic tradition or traditions, for a language to pass from one system at one point in time to another system at a later point. [Watkins, 1973]

Given that the goal of historical linguistics is not the use of the historical comparitive method but, as Watkins describes, uncovering "the linguistic history of known languages," [Watkins, 1973], linguists ought to be open to entertaining any field of research which has additional information to bear on a subject. The mathematical modelling of linguistics, particularly through statistical means, can be simply that: another way of shedding additional light from a different perspective on the data that we have before us as linguists.

Additionally, Calvert Watkins described the models the comparative method produces this way:

> We must not forget, of course, that the reconstruction, the postulated grammar which is arbitrarily considered the initial point in the historical [comparative] linguistic process, is an artifact reflecting the contemporary state of intellectual development. As such it is subject to change, just as all intellectual artifacts or scientific propositions are. . . . This mutability applies also to the model of kinship relations among a set of languages, the configuration of the family tree, which may also be modified—like any scientific proposition—by new data. [Watkins, 1973]

While averring the legitimacy of the reconstructions created by the historical comparative method as valid scientific models, Watkins is clearly stating that the constructions of the historical comparative method are simply a tool in the historical linguist's tool belt (though admittedly the predominant tool for the last 200 years). Additionally, Watkins clearly states that new scientific insight could certainly lead to a revision of the model of language relationships produced by the army of linguists that has been toiling away with the historical comparative method for centuries. After the spirit of inquiry that Watkins describes above, other methods of modelling language history ought to be quite at home in the historical field, since they simply add additional data points to the models that historical linguists have already constructed.

These mathematical and statistical methods also offer a few additional benefits over the traditional historical comparative method. First, because the historical comparative method involves the linguist creating reconstructions, the manner and quality of the reconstructions is often thrown into doubt between linguists who have a competing view point. This subjectivity of the historical comparative method has led to long standing disputes between linguists working in the same field who simply see the data in different ways. By using an algorithmic or statistical method to reconstruct relationships between languages instead of an "arbitrarily considered" (according to Watkins) starting point to reconstruct relationships, arguments about relationships can be based on quantifiable evidence instead of qualitative assertions.

Another advantage that mathematical and statistical methods offer comes about because of the age of "Big Data" that we live in. In a post-Google world, we have gigantic data sets available to us as scientists that would have been previously unfathomable. As large linguistic data sets continue to proliferate, it becomes nearly impossible for one linguist to perform the individual close analyses that the historical comparative method requires across all available data by hand. These mathematical and algorithmic methods, when performed computationally, allow linguists to be able to consider large data sets in their entirety during linguistic analysis in a way that wouldn't

have been possible, even a few years ago.

Perhaps in part because of these advantages that statistical and mathematical methods bestow, in recent years there has been a rekindling of interest in the use of statistical methods in the pursuit of historical linguistics. During the 50 or so years that linguists attempted to distance the field from the use of flawed statistical methods such as glottochronology, biologists and mathematicians have developed statistical and computational methods that are quite useful for answering specific questions about how species are related to one another and the specifics of those relationships (including the modelling of the timing of splits between species and the modelling of their family tree relationships). These methods and the questions they answer have clear parallels in linguistics, and recently a number of linguists have begun investigating them, creating a lexicostatistical renaissance in the field of historical linguistics. Just as Watkins suggested might be possible, by using statistics historical linguists are able to help uncover new information which augments and revises the configuration of the established history of known languages.

In this thesis I will look at a new way of generating such knowledge: the ALINE distance between languages. I will explain how the ALINE distance works and compare it with a different distance metric between languages called LDND, which has been formulated by a research group called the ASJP Project. By examining how these two metrics perform on the same set of data, I will show that the ALINE distance is comparable to LDND, which is probably the best studied automatic language distance metric currently available.

The rest of the thesis, then, will proceed as follows. In chapter two, I will examine the state of the art of current methods for determining the distance between two languages. Then, in chapter three, I will present the methodology that I use for building family trees and the methodology that I have used for comparing two different distance metrics against each other. In chapter four I will show how well ASJP's distance metric and the ALINE distance metric work comparatively and show that ALINE is an equally valid distance metric. In chapter 5, I will summarize my thesis and

suggest possible areas for future research.

Chapter 2

The State of the Art

One of the first questions to ask when setting out to use a mathematical model is: "What am I trying to model and why?" Asking this question helps frame how the model should work in order to better understand the real-world process under examination. For a historical linguist using lexicostatistical methods, one answer to this question might be: "I'm trying to model the relationships between languages. Once I've better understood the mathematics of language relationships, I'll understand better how one language becomes another." Obviously there are various aspects of the relationships between languages that might be investigated with such a model.

The specific aspect that is currently most studied in historical lexicostatistics is how to automatically classify languages into groups in a way that approximates our current understanding of language relationships. In what is currently the most common approach to this problem, two things are required in order to generate such automatic family trees: a distance matrix (which records the distances between individual languages), and a method for generating a tree from a distance matrix. There are a number of standard algorithms for generating trees from such a distance matrix, so most research in historical lexicostatistics has concentrated on the first problem: how do you come up with a number that represents how different two languages are from each other? To situate the work that I have done, I will examine three different approaches to this problem of numerically representing language relatedness.

The first approach has been pioneered by McMahon and McMahon [McMahon and McMahon, 2005].

They use a database of words constructed by Dyen, Kruskal and Black which is a large set of Swadesh lists that have been generated for the Indo-European languages. Encoded in the database for each set of two lexical items with the same meaning is a column which says whether or not the words are cognate. This is pre-determined by hand, based on the judgments of historical linguists and encoded in the database. For each pair of languages in the database, a distance number between the pair of languages is generated based on the percentage of words in the two lists that are cognates. McMahon and McMahon then further refine the items in question by using a set of criteria for determining which lexical items are less likely to be borrowed through contact. This allows them to create two sublists, one which is highly likely to be affected by borrowing and therefore less stable over the long run, and one which is more likely to be unaffected by borrowing and therefore more stable over the long run. They recompute distance matrices based only on the stable sublists from the Dyen, Kruskal and Black database. The result matrices are then turned into trees using off-the-shelf biological tree-building algorithms [McMahon and McMahon, 2005].

A second approach has been proposed by a group of researchers called the ASJP Project [Holman et al., 2008]. After having collected Swadesh lists for more than 3500 languages from all over the world, the ASJP Project devised a procedure which allows them to create trees for these languages using an automated process to generate the distance matrices. For each set of two languages in the whole pool of available languages, the individual items of the Swadesh list are compared using the following process: two words with the same meaning are judged as similar based on a particular distance metric called the Levenshtein distancewhich measures the number of edits used to turn a word from the first language into a word from the second [Holman et al., 2008].

The Levenshtein distance between two words is defined as the minimum number of insertions and deletions (also called insdels) and substitutions that it takes to turn one word into the other [Levenshtein, 1966]. To compute the Levenshtein difference between *cat* and *car* as in Table 2.1 above, we create a distance matrix which we will use to determine the minimal difference between

		c	a	r
	0	1	2	3
c	1			
a	2			
t	3			

		c	a	r
	0	1	2	3
c	1	0	1	2
a	2	1	0	2
t	3	2	1	1

Table 2.1: On the left is an empty distance matrix waiting to be filled in by the Levenshtein algorithm. On the right is a trivial Levenshtein distance matrix for computing the difference between *cat* and *car*. The number at the bottom right of the filled-in matrix gives a distance of *1* since substituting *r* for *t* is all that's necessary to transform *cat* into *car*.

the two words, as on the right in Table 2.1 above. Matrix cells are represented by two pairs of numbers which represent the row followed by the column of an individual cell. Each cell in the matrix row i, column j represents the minimum number of insdels and substitutions needed to transform the string on the left of the matrix up to character i into the the string on top of the matrix up to character j.

For example, let's call the upper left cell (which is blank in the matrices in Table 2.1) cell $0, 0$ Since there are no characters of either string represented by cell $0, 0$ it has a value of 0. This means that cell $2, 1$ represents the number of insdels and substitutions it would take to transform the *ca* from *cat* into the *c* from *car*. Since *ca* is just one letter different from *c* (the *a* can be inserted into *c* or deleted from *ca* to make the strings equal to each other) the Levenshtein distance value of cell $2, 1$ is 1. Similarly, cell $3, 2$ represents the minimum number of insdels and substitutions that are required to turn *cat* into the *ca* from *car*. Since *t* can simply be removed from *cat* to make it *ca*, and vice versa, this number is 1, as is the value in $3, 2$.

The Levenshtein distance algorithm visits every blank cell in the matrix on the left in Table 2.1 on page 11, from the top to the bottom, and from left to right. At each step, the algorithm looks at the characters from each of the two strings in question that are indicated by a given cell. If the two characters are equivalent, it sets the value of the cell in question to be equal to the value of the cell to its upper-left. If they are different, the algorithm adds one to the lowest value from all

three surrounding cells that have already been populated. When the algorithm finishes, the cell in the bottom righthand corner represents the total number of insertions, deletions or substitutions necessary to turn one string into the other.

The ASJP Project's methodology uses a modified version of the Levenshtein distance which they call the LDND, which stands for Levenshtein Distance Normalized and Divided. They start by calculating the Levenshtein distance between all of the word pairs of two languages. Each word is phonetically encoded, so the Levenshtein distance is comparing phonemes with each other, not written characters. ASJP then normalizes the distance between any two words by dividing by the length of the longest of the two words compared. Finally this value is divided by a number which represents the amount of natural phonetic overlap between the two languages. This value is computed by taking the average Levenshtein distance between all of the possible combinations of word pairs which don't have the same meaning and averaging this into one number. This last step is performed to average out any chance phonetic overlap between the two languages in question [Wichmann et al., 2010]. This resulting number is subtracted from 100% and with this percentage, the ASJP Project obtains a distance for each pair of languages in their database [Holman et al., 2008]. At this point, they also use off-the-shelf biological tree-building algorithms.

A third approach to determining the distance between two languages has recently been examined by Downey et al. [Downey et al., 2008]. They use an algorithm called ALINE, developed by Grzegorz Kondrak, which actually looks at the phonetic distinctive features underlying individual words in order to calculate distance matrices for those languages. Since this is the algorithm examined by my thesis, I will explore its provenance in some detail.

In his original description of the ALINE algorithm, Grzegorz Kondrak surveys a different set of algorithms for determining the phonetic distance between two words [Kondrak, 2000]. The first algorithm that Kondrak describes was proposed by Michael Covington. It first groups phonemes into three classes: consonants, vowels and glides. Convington's algorithm then assigns a distance

to each of the possible differences in classes between two phonemes. Additionally, deletions and insertions of phonemes between the two words being compared are also assigned penalty scores which are added to the overall distance measurement for the two words.

Kondrak also briefly describes two other distance metrics given by Gildea and Jurafsky, and Nerbonne and Heeringa respectively. Rather than just using a single feature or a phoneme class, both of these distance metrics use binary feature vectors to represent the words being compared. A distance metric called the Hamming distance is used in both of these metrics to penalize a substitution from one phoneme to another, while insertions and deletions of phonemes are penalized somewhat arbitrarily by both.

In contrast to these metrics, Kondrak proposes ALINE [Kondrak, 2002]. He shows that ALINE performs well under a variety of circumstances. ALINE works over multivalued feature vectors representing the words in question; that is, where Gildea and Jurafsky's and Nerbonne and Heeringa's methods both use feature values that are either on or off, Kondrak's ALINE has a varying fraction between 0 and 1 for the multiple values of each of the features considered (see Table 2.2 on page 14 for a list of features and their values). Additionally, ALINE, rather than computing the distance between two words, computes the words' similarities, giving higher scores for words which are more similar and lower scores for words which are less similar [Kondrak, 2000].

In order to compute the similarity between two words, such as *cat* and *car* as described above for the Levenshtein distance, ALINE first decomposes them into a set of feature value vectors, one vector for each phoneme in the word. Note that since ALINE is considering the phonetic material in the words, it has an encoding scheme for translating regular letters into phonemes, similar to that used by ASJP. For example, in the ALINE representation of *cat* the letter *F* is added after the *a* so that the string becomes *caFt*, to represent that the *a* is a front vowel. The *a* in *car* doesn't have this modifier since it is not a front vowel.

ALINE creates a similarity matrix which is very much like the distance matrix used by the

Feature Name	Value Name	Value
place	bilabial	100
	labiodental	95
	dental	90
	alveolar	85
	retroflex	80
	palato-alveolar	75
	palatal	70
	velar	60
	uvular	50
	pharyngeal	30
	glottal	10
manner	stop	100
	affricate	90
	fricative	80
	approximant	60
	trill	50
	vowel	40
	high vowel	40
	mid vowel	20
	low vowel	0
high	high	100
	mid	50
	low	0
back	front	100
	central	50
	back	0

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

Operation	Cost
Skip	-1000
Substitution	3500
Expansion	4500
Vowel	1000

Table 2.2: On the left are the multivalued features and their values as defined by ALINE, taken originally from Ladefoged [Kondrak, 2002]. PyAline adds *trill* as a value for the feature *manner* to the original set from ALINE. This is because the ASJP dataset encodes trills. ALINE also uses the following binary features for each phoneme: syllabic, voice, nasal, lateral, aspirated, retroflex, long, round. On the right, the salience of each feature and the penalty/reward for each operation that ALINE allows between strings. The values are adjusted from those originally reported by [Kondrak, 2002] so that they match the behavior of his implementation.

		c	a	r
	0	0	0	0
c	0	35	25	15
aF	0	25	47.5	37.5
t	0	30	37.5	40.5

Table 2.3: On the left is an empty distance matrix waiting to be filled in by ALINE. On the right is a trivial Levenshtein distance matrix for computing the difference between *caFt* and *car* which are the phonetic encodings for *cat* and *car* using ALINE's phonetic encoding scheme. The largest value in the matrix is the overall similarity between the two words. In this case it's the middle of the matrix that has this value, which is 47.5.

Levenshtein distance, as can be seen in Table 2.3 above. Each cell in ALINE's similarity matrix represents how similar a particular set of phoenemes is under the optimal alignment up to that point. ALINE considers the phonetic similarity of insdels and substitutions, much like the Levenshtein distance. It also adds phonetic compressions and expansions to the processes it considers to be relevant at each step. This allows it to consider a case such as Latin's *dictum* turning into the Italian *detto* where the single long *t* phoneme represented by the *tt* should be linked with both the *c* and *t* phonemes in the Latin word, rather than simply deleting one of them from the comparison [Kondrak, 2002].

In table 2.3, cell $1, 1$ with a value of 35, represents the phonetic similarity of *c* from *caFt* and *c* from *car* [1]. Similarly, in cell $1, 2$ we see the phonetic similarity between *caF* from *caFt* and *c* from *car*. Since ALINE is a similarity algorithm, and making this change requires an insertion or a deletion, this is penalized and the score for the similarity between *caF* and *c* is smaller than the score between *c* and *c*.

ALINE fills in an empty matrix in the same order as the Levenshtein distance algorithm described above does, from top to bottom, left to right. At each step, it examines similarity value of each of the three possible changes noted above: inserting or deleting one of the phonetic segments

[1] Kondrak points out that since ALINE uses a similarity approach instead of a distance approach, the score of identical segments isn't 0, but is instead based on the feature values between them [Kondrak, 2002]

in question; substituting them for one another; or expanding or contracting two segments from one word into the other. It chooses whichever change (or no change) has the maximum similarity and adds it to the previous similarity value which has been computed up to this point. Each type of change is assigned a base score, which is then altered by the amount of phonetic similarity between the phonemes involved in the change (see Table 2.2 on page 14 for the list of features, saliences and base scores). The phonetic similarity is based on the differences between the numerical value of the features for each phoneme, and different features are given different saliences in this calculation. A larger salience means that the feature causes a bigger difference to be calculated between the phonemes in question.

As an example, the similarity between between *b* and *p*, which differ on the voiced feature, has a score of 25, while the similarity between *g* and *p*, which differ on more than one feature, is scored much lower with a value of 9. When the matrix is filled with the appropriate summation of these similarity scores, the cell in the matrix with the largest value represents the phonetic similarity of the two words in question [Kondrak, 2002].

Downey et al. [Downey et al., 2008] develop a distance metric between languages using Kondrak's ALINE for determining the distance between two words, and then use the average ALINE distance between languages to compute distance matrices between groups of languages. In order to normalize the ALINE similarity score and turn it into a distance metric, they subtract 2 times the ALINE similarity divided by the sum of comparing each word with itself since that gives the most similarity. As two words become more similar, the sum gets closer to 1. To turn this number into a distance metric instead of a similarity metric, they subtract the value from 1. Thus, as two words become closer the distance between them approaches 0. The equation looks like this:

$$d_{ALINE} = 1 - \frac{similarity(word1, word2)}{similarity(word1, word1) + similarity(word2 + word2)}$$

Downey et al. then average the d_{ALINE} for all the pairs of semantically equivalent words between two languages to come up with a distance measure between two languages.

The ALINE distance between languages appears to have a couple of advantages over the other two metrics used by McMahon and McMahon, and the ASJP Project. In the case of McMahon and McMahon, using ALINE replaces the work done by linguists, eliminating some subjectivity in the results and making cognancy judgments much more analog. Rather than giving an absolute "Yes, these two are cognates" or "No, these two are not cognates," it says, "These two are 10% cognates." In the case of the modified Levenshtein distance that the ASJP Project uses, ALINE appears to have an advantage because it actually takes into consideration the underlying distinctive feature values in a particular set of phonemes, allowing phonemes like *t* and *d* to be more similar mathematically than, say, *t* and *m*. As previously mentioned, the Levenshtein distance that the ASJP Project uses would simply conclude *t* and *d* are not equal rather than saying they're mostly equal, as ALINE does.

While the ALINE distance seems to be a promising new entrant in this field, there has been so far no objective way to measure one method against another. In the next chapter I will describe how I have used the ALINE distance to measure the distance between the various Swadesh lists created and gathered by the ASJP Project. I also discuss how I compared the trees generated by the ASJP Project's program with those which the ALINE distance generates[2].

[2]Unfortunately, since the McMahon and McMahon method relies on expert linguists to decide if two words are cognate with one another, it's not possible (or practical) to compare it with the other two, since they're capable of looking at a much larger set of language data without human intervention. Therefore I was unable to compare the McMahon and McMahon method with the other two for accuracy.

Chapter 3

A Methodology

To summarize the discussion in the last chapter, researchers have been attempting to use mathematics to model the relationships between languages. Once we better understand the mathematics of language relationships, we'll understand better how one language becomes another. Of course, one model would suffice for answering this question, but, as we have seen, multiple models now exist for defining the distance between words and consequently between languages. How can we determine if one language distance metric is more accurate than another? The most intuitive way to do this would be to run them on the same set of data, generate a set of trees, and compare the trees generated by the model against trees created by experts.

Since the ALINE distance is a relative newcomer to this field, it has not been tried on as wide a data set as some of the other metrics, particularly the ASJP distance metric. For the purposes of this thesis, I set out to determine how well the ALINE distance worked for generating language family trees in comparison with the ASJP distance metric.

To measure the ALINE distance against the ASJP Project's Levenshtein-based distance, a set of languages could be fed into both distance metrics. Then the trees generated from both metrics' output can be compared with a set of trees generated by linguists to see whether ASJP or ALINE is closer.

As I began my research, I knew I would need an easily manipulable implementation of the ALINE algorithm, and the existing C++ implementation was not conducive to this. So I reimple-

mented the algorithm in Python and called the resulting program PyAline[1]. I also needed data so that I could compare the two algorithms on a wide variety of different types of languages.

I approached the ASJP Project with a request to use their data, and was generously granted both access and eventually membership in the consortium. As I examined the data, though, I realized I had a second problem: the ASJP Project's data uses its own encoding format to describe phonemes, as does ALINE, and the two encoding formats differ greatly. It was at this point that I was able to first make use of my reimplementation of ALINE in Python. It was straightforward to simply swap out the input reading portion of the PyAline implementation and create feature vectors based on the ASJP encoding scheme instead of the ALINE encoding scheme. Since I could use the same encoding scheme for both the ALINE and ASJP algorithms, I could take the same input and compare the trees generated based on the output from both algorithms. I was now ready to compare apples to apples.

As I began running large sets of languages through both programs, it quickly became apparent that running ALINE's dynamic programming algorithm across varying feature sets was significantly slower than ASJP's more simplistic Levenshtein distance. This is in part because in the step where a Levenshtein distance compares to see if two phonemes are equal and assigns a binary 1 or 0 to this comparison, ALINE compares each feature between the two characters and then assigns a more analog weight to the differences it finds. Where ASJP was doing one comparison per phoneme pair, I was comparing across 15 individual features for that same phoneme pair. Data

[1]PyAline can be downloaded at http://sourceforge.net/projects/pyaline/files/pyaline-0.0.1.tgz/download

During this reimplementation in Python I discovered a few minor issues in the description of the algorithm by Kondrak. While comparing PyAline's alignments with ALINE's, I had trouble getting the alignments to match. I eventually discovered that the penalty factors that the algorithm gives to deletion, substitution, expansion and skipping were all off by a factor of 100 as described in Kondrak's PhD thesis. Once I adjusted them accordingly (multiplying each by 100), I was able to get PyAline's alignments to exactly match those generated by Kondrak's implementation of ALINE. Table 2.2 on page 14 reflects the corrected values from Kondrak's original description.

runs for one large set of languages with PyAline were taking more than a day and a half to finish on a MacBook Pro with a 2.53 GHz Intel Core 2 Duo and 4GB of RAM, which at the time of this writing is a reasonably fast machine.

To speed up the generation of results I was able to once again take advantage of the modularity of the PyAline implementation of ALINE in order to parallelize the generation of distance matrices between languages. I did this by generating a set of wrapper programs around PyAline that was able to parallelize the most costly portions of the algorithm by running each calculation between two sets of language in parallel on an Apache Hadoop cluster.

Apache Hadoop is an open source system which runs large processing jobs across huge amounts of data. It implements a massively parallel programming paradigm developed by Google called MapReduce. The first part, Map, applies an operation in parallel to every member in a list of items. Reduce, the second step, performs an operation which normally aggregates the results of the Map step into a final data structure. Google's original purpose was to create an index of the entire web by processing websites in parallel across multiple computers, each with multiple processors using a Map, and then using the Reduce step to combine those processed results into a central index.

Hadoop is an incredibly useful tool for processing and better understanding large amounts of data. So, naturally, when confronting the large amount of linguistic data I had from the ASJP Project, I set out to implement PyAline in such away that Hadoop could process it via MapReduce.

In order to create a language family tree from Swadesh lists, we must compare each set of 2 languages in the data in question to one another and obtain a distance for each pair. This was a natural Map step. Therefore, I created a script which would generate a list of all pairs of the languages, and fed this into Hadoop with a Map function that would take a pair of languages, and output the ALINE distance between those two languages after calculating it. The Reduce step then simply became the creation of a matrix of the resulting distances, putting each distance into the right place. Since the ALINE distance is symmetric (i.e. $d_{ALINE}(language1, language2) ==$

$d_{ALINE}(language2, language1)$) I don't need to fill up every square in a distance matrix and compare each set of languages twice. Therefore, I implemented this speed up in PyAline and the Hadoop implementation of PyAline, as well, only computing half the distance matrix for each set of languages, since the other half was exactly the same.

Once I had a reasonable implementation of PyAline running on top of Hadoop, I used a generous set of cluster computing grants from Amazon.com. Amazon has a large set of computers that it rents out on an hourly basis[2]. Using 20 machines of Amazon's at a time, and running PyAline in a Map Reduce mode I reduced the processing time down to an hour from several days for some of the larger language groups.

Having obtained distance matrices for both the ASJP distance and PyAline distance I needed to build a set of trees from those distance matrices. There are several algorithms used in the bioinformatics community which generate a tree based on a distance matrix. The two that I chose to experiment with for the purposes of this thesis are the Unweighted Pair Group Method with Arithmetic Mean (UPGMA) method and the Neighbor Joining (NJ) method, which are the same two that Downey et al. used in their preliminary experiments with the ALINE distance.

Both methods start with a distance matrix which represents a collection of different items and the distances between each pair of items (as I generated above using PyAline and ASJP). They then examine those items and find the two which are closest together based on some distance function, and then combine the two items into a single bifurcation on a tree, though they differ in how they choose which items are the closest together.

As part of its first step, UPGMA picks the two items which have the closest distance to each other in the distance matrix. It makes a new node between them in the tree it's creating and combines them into a single item in the distance matrix. It calculates the distance between the new

[2]Amazon give grants of computing time on these cluster computers to students who are learning to use Hadoop and other cluster computing tools. They also give grants to other researchers to help them have computing resources on which to run their research programs.

item in the distance matrix and the rest of the existing items by averaging the distance between the individual items contained in the new item and the other items in the collection under consideration. This process is repeated until every item in the distance matrix is part of the tree and all the created nodes are connected to each other. See Figure 3.1 on page 23 for an example of how UPGMA works.

The are two big differences between UPGMA and NJ. The first is that NJ begins with a tree structure where all of the items in the distance matrix are connected to a single central node. The other is that NJ computes a second matrix called the Q matrix based on the distance matrix. There are several different equations for generating the Q matrix but they all produce the same trees, because they all seek to minimize the lengths of the branches of the resulting tree when two nodes are combined. So, whereas UPGMA uses the smallest distance between two nodes to pick which two nodes to choose next, NJ picks the two nodes whose resulting tree has the smallest maximum branch length possible. One popular equation for computing the Q value between two nodes which we'll call i and j is this:

$$Q(i,j) = (numberOfItems - 2) * d(i,j) - \sum_{k=1}^{numberOfItems} d(i,k) - \sum_{k=1}^{numberOfItems} d(j,k)$$

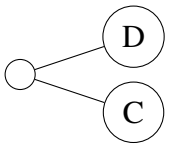
This equation sums over the distance between i and every other node in the distance matrix and j and every other node in the distance matrix. Doing so, it approximates the length of the resulting tree were these nodes to be combined [Gascuel and Steel, 2006].

Once the nodes with the smallest Q value are selected, they're combined with an extra node, which is then connected to the central node that we started with. For the purposes of these equations, let's call this new node u and the nodes which it connects f and g. The distance between this new node u and any node k that's already in the distance matrix can be calculated as follows:

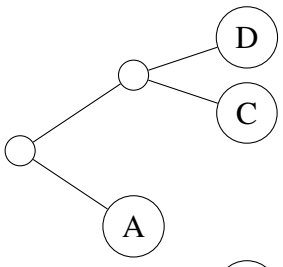
$$d(u,k) = \frac{1}{2}[d(f,k) - d(f,u)] + \frac{1}{2}[d(g,k) - d(g,u)]$$

Clearly this equation relies on the distance between our new node u and the nodes it comebined f

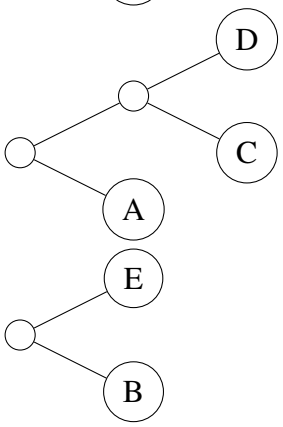
Step 1	A	B	C	D	E
A	x				
B	5	x			
C	2	7	x		
D	3	4	1	x	
E	9	3	4	2	x



Step 2	A	B	CD	E
A	x			
B	5	x		
CD	2.5	5.5	x	
E	9	3	3	x



Step 3	ACD	B	E
ACD	x		
B	$5.\overline{3}$	x	
E	5	3	x



Step 4	ACD	BE
ACD	x	
BE	$5.1\overline{6}$	x

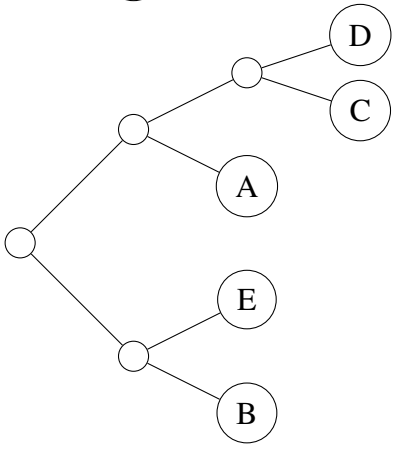


Figure 3.1: This graph represents how UPGMA would construct a tree from the above distance matrix. **Step 1**: The smallest distance is between items C and D which have a distance of one. These two items are combined into one node CD, which has a distance of $d(A,C)+d(A,D)/2 = (2+3)/2 = 2.5$ from A, a distance of $d(B,C)+d(B,D)/2 = (7+4)/2 = 5.5$ from B and a distance of $(d(E,C)+d(E,D)/2 = (4+2)/2 = 3$. **Step 2**: Since the average distance of CD to A is smaller than the distance to B or to E, add A as the next node to the tree, and create a column ACD. **Step 3**: The distance between B and E is smallest, so they are joined into a node. **Step 4**: Finally, there are only two nodes left, BE and ACD and they are joined.

and g. This distance can be calculated thusly for f (we just need to replace f with g to calculate the distance for the other node):

$$d(f,u) = \frac{1}{2}d(f,g) + \frac{1}{(2(numberOfItems) - 2)}\left[\sum_{k=1}^{numberOfItems} d(i,k) - \sum_{k=1}^{numberOfItems} d(j,k)\right]$$

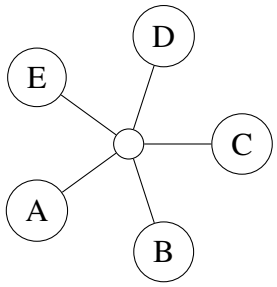
These equations calculate a new distance matrix for the newly attached node to the tree. This process is repeated until there are 3 items left in the distance matrix. Since all the items start connected to the central node, they're always part of the tree, so when there are 3 items left the tree has been appropriately built. Since NJ is trying to choose the nodes which will lead to the smallest resulting tree length at each step, the resulting tree should be pretty close to the smallest tree available for the given items. See Figure 3.2 on page 25 for an example of how NJ works [Gascuel and Steel, 2006].

I chose the NJ implementation that is provided by the *ape* package in the R statistical language system [R Development Core Team, 2010]. I also used R's default UPGMA implementation. After running UPGMA and NJ on both the ASJP and ALINE data, I had two sets of automatically generated trees for each tree-building method. In order to tell which set of trees (and consequently which distance metric) was more accurate, I needed a way to compare trees to each other. I realized that if I had a set of expert trees, I could compare the distance between each of my sets of generated trees and the expert trees, and whichever methodology got closer to the expert trees could be considered better.

I was able to construct two sets of such expert trees. Ethnologue [Lewis, 2009] has a set of language information (including classifications) about many of the languages which are represented in the ASJP database. Luckily, the ASJP data contains the standard ISO three letter code for each language that has a Swadesh list, and each language on the online version of Ethnologue is accessible based on its ISO code. I was able to write a program which took this code and visited the Ethnologue website and parsed the family lineage for each language from the web page and

Step 0

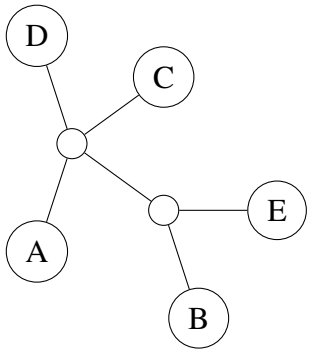
	A	B	C	D	E
A	x				
B	5	x			
C	2	7	x		
D	3	4	1	x	
E	9	3	4	2	x



Step 1

	A	B	C	D	E
A	x				
B	5	x			
C	2	7	x		
D	3	4	1	x	
E	9	3	4	2	x

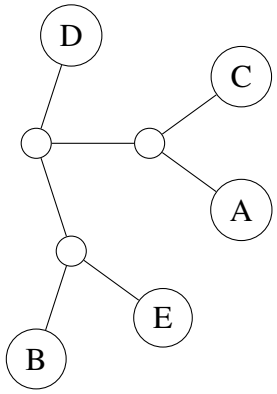
Qs	A	B	C	D	E
A	∞				
B	-23	∞			
C	-27	-12	∞		
D	-20	-17	-21	∞	
E	-10	-28	-20	-22	∞



Step 2

	A	C	D	BE
A	0	2	3	5.5
C	2	0	1	4
D	3	1	0	1.5
BE	5.5	4	1.5	0

Qs	A	C	D	BE
A	∞			
C	-13.5	∞		
D	-10	-10.5	∞	
BE	-10.5	-10	-13.5	∞



Step 3

	D	BE	AC
D	0.0	1.50	1.00
BE	1.5	0.00	3.75
AC	1.0	3.75	0.00

See graph from Step 2

Figure 3.2: **Step 0**: NJ starts by linking all the items in the distance matrix to one central node. **Step 1**: A Q matrix is constructed, representing the length of the total tree if two nodes were to be linked together as described above. The two nodes with the lowest Q are joined with a linking node. A linking node replaces the two nodes on the central node. B and E are chosen. **Step 2**: A distance matrix is created including the new node. Q values are generated for it and the same process is repeated to find the next branch of the tree. There's a tie between nodes A and C and node D and the new BE node. A and C are arbitrarily chosen and joined. **Step 3** NJ stops when there are 3 items in the distance matrix, so the process is now complete.

then put this information back together into a set of expert trees from Ethnologue[3]. A second source of expert trees was found encoded along with the ASJP dataset. Each language in the ASJP dataset is given a classification based on a modified version of the Ethnologue dataset. In some cases, extra groupings are added that would most likely be in Ethnologue if the languages had been known at the time of its construction. In other cases multiple languages are given the same 3 letter ISO code because there are Swadesh lists for languages in the ASJP dataset which Ethnologue hasn't classified yet. In these cases the strict Ethnologue expert tree described above will place all the languages with the same ISO code in the same branch, while the ASJP modified Ethnologue classification might place them differently. Both sets of trees are not binary, which means that a mother language or group can have multiple children at each level. This is an important distinction because the trees which I generated using NJ and UPGMA were binary trees.

Since the languages that are classified by the strict Ethnologue set are a slightly different set than those classified by the modified Ethnologue dataset provided by the ASJP project, I had to remove some languages from consideration by ALINE and ASJP in the strict Ethnologue tree classification experiment. I made the data preparation step which generated Ethnologue trees also generate the data sets for ASJP and ALINE input so that I could run one program on several inputs and generate a synchronized set of inputs for ASJP and ALINE at the same time as I created expert trees to compare them against.

After passing the input languages into ASJP and PyAline to get a distance matrix, and using neighbor joining and UPGMA to create trees from the distance matrices, I had a set of trees for each methodology. I needed a way to determine whether one set of trees was more accurate than the other at approximating the expert trees. The *phangorn* package for R provides a

[3]This script is publicly available under an open source license and can be found at the ASJP website: http://www.eva.mpg.de/%7Ewichmann/process_asjp-0.0.1.zip

The trees can be found on the ASJP website as well at the following urls: http://www.eva.mpg.de/%7Ewichmann/Strict_Ethnologue_family_trees.zip and http://www.eva.mpg.de/%7Ewichmann/Modified_Ethnologue_family_trees.zip

way of measuring tree distances using the popular Robinson-Foulds distance metric between trees [Steel and Penny, 1993]. To calculate the Robinson-Foulds metric between two trees, the number of different ways each tree can be cut in half (called a split, or a bipartition) is calculated. Then the two sets of splits are compared to each other (see Figure 3.3 on page 28 for an example using the trees described earlier). The value of the metric between two trees is twice the number of splits that are different between the trees [Steel and Penny, 1993]. So, to determine whether one set of trees was ore accurate than the other I simply calculated the Robinson-Foulds distance between the ASJP generated trees and the expert trees and then calculated the Robinson-Foulds distance between the PyAline generated trees and the expert trees. At this point I had an objective, quantifiable measure of the relative accuracy of the two language distance metrics.

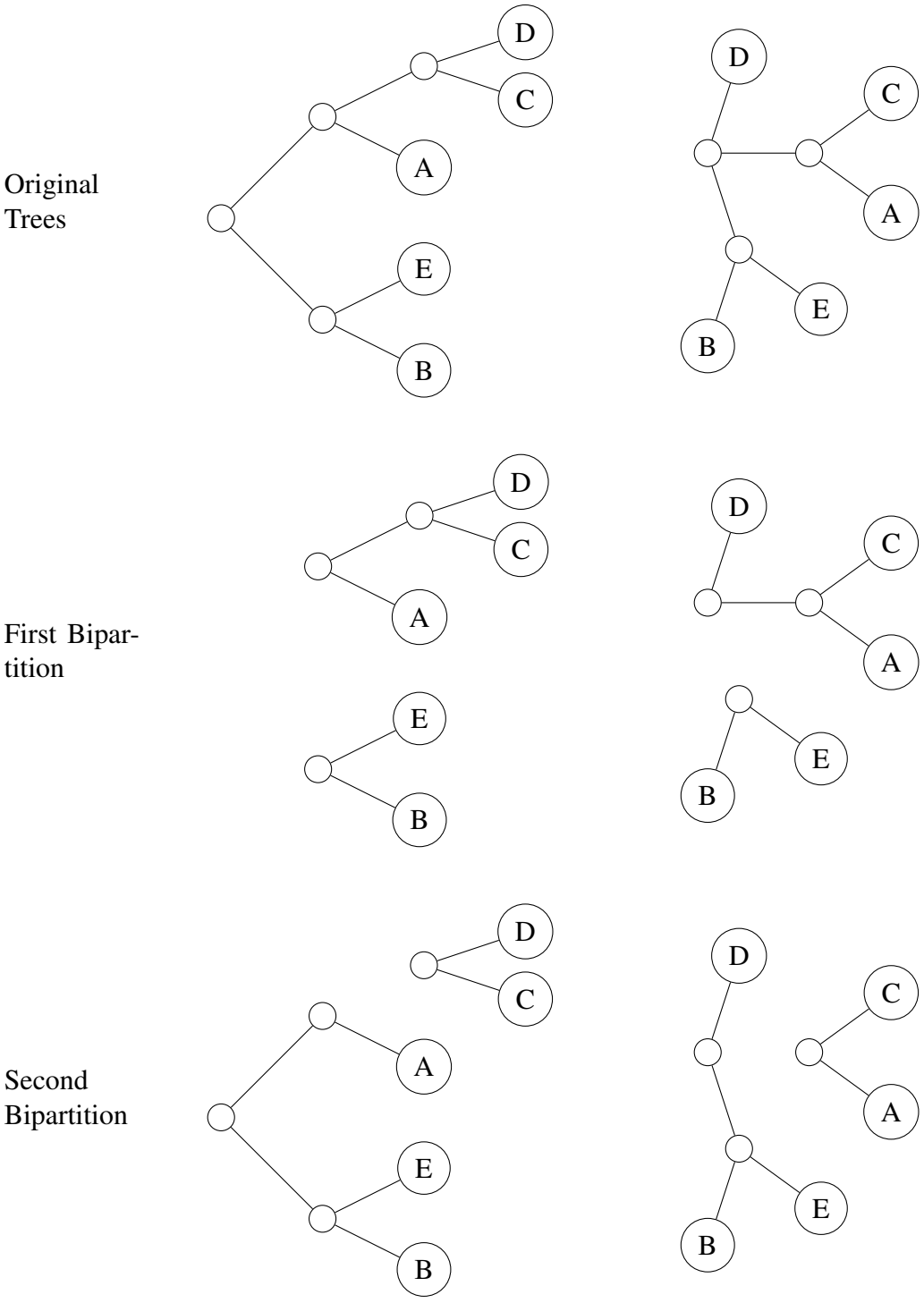


Figure 3.3: RF distance between two trees is twice the number of splits that are different between the two trees. Splits are created by removing an internal edge (one which doesn't have a leaf node attached to it). In the top example we see the original trees again. The second level shows the removal of an internal edge on both trees which isolates nodes B and E on one half and A C and D on the other. Though A C and D are configured differently, these bipartitions are considered equivalent. The third level shows the removal of the other internal edge on both trees. This results in a different bipartition for each tree. This means that there is 1 split which is different between the two trees. The distance is $1 * 2 = 2$ since there are no other internal edges to be removed [Steel and Penny, 1993].

Chapter 4

Results

In this chapter we will see the fruits of the methodology outlined in the previous chapter. I will examine the overall results of both the ALINE distance and ASJP's results for a wide ranging set of language families, using two sets of expert trees, one based on the ASJP Project's modified Ethnologue classification and the other based on the strict Ethnologue classification [Lewis, 2009].

As is seen in Table 4.1 on page 29 below, the modified Ethnologue classification data set includes 58 language families comprising Swadesh lists for 3878 languages in total. The largest group, the Austronesian family, is made up of 845 languages. The smallest group is comprised of 7 languages from the Chukotko-Kamchatkan family of northeastern Siberia. The mean number of languages in each group is 66.86, with a standard deviation of 138.33. Given the wide standard deviation, it's clear there is a wide variety of sizes of language families. For the purposes of having

Modified Ethnologue Data Set	
Language Families	58
Total Languages	3878
Largest Family Size	845
Smallest Family Size	7
Mean # of Languages	66.86
Standard Deviation	138.33

Table 4.1: A description of the basic statistics concerning the language families in the modified Ethnologue data set.

meaningful data, I removed language families for which there was no expert tree (all the languages were on the same level of the tree). Since both tree-building algorithms were incapable of constructing a structure other than a tree, the Robinson-Foulds (RF) distances between the constructed trees and the expert non-classification would have been essentially meaningless.

A quick glance at the first two columns of Table 4.2 on page 31 shows that PyAline and ASJP return similar results for each of the language groups in consideration here. Of the 58 language families in consideration in this experiment, there are 12 families (21%) for which PyAline and NJ produce a tree which is closer to the expert tree than ASJP, 15 families (26%) where ASJP and NJ produce a tree which is closer to the expert tree than PyAline, and 31 families (53%) where ASJP and PyAline when combined with NJ produce trees which are the same distance from the expert tree. This is interesting, because there are only 8 families of the 58 for which ASJP and PyAline combined with NJ produce the same tree (see the distance between the ASJP- and PyAline-generated trees in the third column; those with distance 0 are equivalent trees). This means that nearly 40% of the time, PyAline and ASJP are creating distance matrices which yield different trees using NJ but which are the same distance from the expert trees provided by the modified Ethnologue data.

What this large collection of trees which are the same RF distance from the expert tree while not being the same tree indicates is that ASJP and PyAline are working in such a way that they both generate different suggestions for how a group of languages are related to one another, and both of those suggestions are objectively equivalent, though they are structurally different. An expert in the language family might prefer one structure over the other, but the RF distance metric is incapable of doing so.

The mean difference between the RF distances of the trees generated by each is 1.76 with a standard deviation of 3.16, which means that in 95% of the language families examined, the automatically generated trees have RF distances to the expert tree that are within 8.08 points of

Language Family	ALINE NJ	ASJP NJ	ALINE-ASJP NJ	ALINE UPGMA	ASJP UPGMA	ALINE-ASJP UPGMA
Afro-Asiatic	374	376	148	378	372	118
Algic	46	46	28	46	48	26
Altaic	136	132	58	140	134	80
Austronesian	1488	1470	660	1490	1474	750
Arawakan	88	88	26	90	88	36
Austro-Asiatic	64	60	30	66	66	36
Australian	314	312	132	308	314	176
Border	24	26	12	26	28	14
Chukotko-Kamchatkan	0	0	0	2	2	0
Carib	28	28	12	32	30	12
Chibchan	34	34	20	36	36	24
Choco	8	8	0	8	8	2
Creole	86	84	32	86	88	40
Dravidian	22	22	8	24	24	10
Eskimo-Aleut	10	10	2	10	10	0
Eleman	4	4	2	4	4	0
Andamanese	14	14	2	16	16	0
Gogodala-Suki	10	10	4	10	10	4
Hmong-Mien	36	36	14	36	36	18
Hokan	32	30	12	34	34	6
Witotoan	4	4	0	4	4	0
Indo-European	400	412	196	408	416	202
Iroquoian	2	2	2	2	2	2
Japonic	4	4	0	4	4	2
Khoisan	22	20	6	20	22	8
Tor-Kwerba	6	6	0	6	6	4
LakesPlain	42	40	12	38	40	12
Ramu-LowerSepik	16	18	2	16	16	2
Macro-Ge	32	34	18	34	36	26
Morehead and Upper Maro Rivers	18	20	4	22	22	2
Mixe-Zoque	14	12	10	14	14	2
Marind	42	42	4	46	42	6
Mayan	152	152	56	152	150	72
Niger-Congo	928	930	458	936	928	470
NorthCaucasian	34	34	14	34	34	16
Na-Dene	26	32	20	32	28	12
Nilo-Saharan	156	162	56	158	164	70
Oto-Manguean	84	90	42	88	90	48
Panoan	30	30	16	32	32	22
Pauwasi	6	6	0	6	6	0
Penutian	20	18	6	20	22	8
Sino-Tibetan	258	258	146	260	262	140
Salishan	12	12	2	12	12	0
Sepik	42	40	20	44	42	24
Siouan	6	4	2	8	4	4
Sko	20	20	4	20	22	4
East Bird's Head-Sentani	10	10	0	10	10	0
Tai-Kadai	104	104	38	104	104	48
Trans-New Guinea	490	482	198	490	476	212
Torricelli	52	50	12	54	54	16
Totonacan	20	20	2	20	20	4
Tucanoan	18	16	4	14	18	8
Tupi	84	84	40	84	84	44
Uto-Aztecan	138	140	70	142	140	82
Uralic	26	28	4	26	26	14
West Bomberai	10	10	2	10	10	0
West Papuan	58	58	12	58	56	22
Yeniseian	4	4	0	4	4	2

Table 4.2: This table shows the Robinson-Foulds distance between the trees generated by PyAline and the expert trees from the modified Ethnologue classification in the ASJP data set. It also shows the RF distance between the expert trees and the ASJP generated trees, and the RF distance between the ASJP and PyAline trees, illustrating how different the two sets of automatically generated trees are from each other.

NJ Results for Modified Ethnologue Data			
PyAline < ASJP	12 (21%)	Mean RF Difference	1.76
ASJP < PyAline	15 (26%)	Std Dev RF Difference	3.16
PyAline == ASJP	31 (53%)		
Same Tree by Both	8 (14%)		

Table 4.3: A table describing the basic outcome of the experiment in which distance matrices created by PyAline and ASJP were turned into trees using Neighbor Joining and compared to modified Ethnologue expert trees

each other. Based on these results, it seems reasonable to conclude that as far as is currently discernible, PyAline and ASJP produce results which are roughly equivalent to each other in terms of how close they match expert trees. These results are summarized in Table 4.3 on page 32.

A similar set of results can be seen for trees generated using UPGMA data as shown in columns 4 and 5 of Table 4.2 on page 31. For UPGMA generated trees, there are 14 families (24%) for which PyAline produces a closer tree than ASJP. There are also 14 families (24%) for which ASJP produces a closer tree to the expert tree than PyAline. The remaining 30 trees (52%) are equivalently distant, using PyAline or ASJP. Only 9 of those trees (16% of the 58 total families, 30% of the equidistant families) are exactly the same (see column 6 in Table 4.2 on page 31 to see how different many of the generated trees are from each other). The mean difference between PyAline and ASJP trees is 2.03 and the standard deviation is 3.27. This means that 95% of the trees generated by ASJP and PyAline are within 8.59 RF distance points of each other. These results are summarized in Table 4.4 on page 33.

Once again, based on these results, it seems like PyAline and ASJP produce fairly equivalent results in terms of how close the trees generated using UPGMA get to modified Ethnologue expert trees. This reinforces the notion that PyAline and ASJP are roughly equivalent language distance metrics, since the trees that are produced from their distance matrices are nearly equivalent a majority of the time.

Interestingly, based on these experiments we can draw another conclusion which is that NJ on

Summary of UPGMA Results for Modified Ethnologue Data			
PyAline < ASJP	14 (24%)	Mean RF Difference	2.03
ASJP < PyAline	14 (24%)	Std Dev RF Difference	3.27
PyAline == ASJP	30 (52%)		
Same Tree by Both	9 (16%)		

Table 4.4: A table describing the basic outcome of the experiment in which distance matrices created by PyAline and ASJP were turned into trees using UPGMA and compared to modified Ethnologue expert trees

Strict Ethnologue-Based Data Set	
Language Families	61
Total Languages	3787
Largest Family Size	826
Smallest Family Size	6
Mean # of Languages	61.08
Standard Deviation	132.10

Table 4.5: A description of the basic statistics concerning the language families in the strict Ethnologue-based data set.

average produces trees which are closer to the modified Ethnologue expert trees than UPGMA. Using the PyAline distance matrices, 26 of the NJ family trees (45%) are closer to the modified Ethnologue expert trees than the UPGMA generated family trees. Only 4 of the UPGMA family trees (7%) are closer than the NJ family trees. The remaining 28 family trees (48%) are equidistant from the modified Ethnologue expert trees whether they are generated by NJ or UPGMA.

Using the ASJP distance matrices, similar numbers are evidenced. There are 26 families (45%) for which NJ produces a tree closer to the expert data than UPGMA, 8 families (14%) for which UPGMA produces a tree closer to the expert data and 24 (41%) families for which UPGMA and ALINE produce trees which are equidistant from the modified Ethnologue expert trees. Based on these two sets of data, it seems clear that NJ is a better tree generation algorithm for the language data such as that provided by the ASJP Project's language database.

I also performed an experiment using a set of expert trees based more strictly on the Ethno-

logue classifications of the ASJP Project's Swadesh lists. The strict Ethnologue-based data set consists of 61 families comprised of 3787 individual languages. The largest family is again the Austronesian family, though in this data set it is made up of 826 languages, a 19 fewer than in the modified Ethnologue data set. The Chukotko-Kamtchatkan family is also the smallest in the strict Ethnologue-based data set, made up of only 6 languages since Ethnologue only knew about 6 of the languages in the data set. This reduction in family size is also evidenced in the mean and standard distribution of the family size in the strict Ethnologue-based data set, with a mean of 61.08 languages per family and a standard deviation of 132.10. See Table 4.5 on page 33 for a summary of these numbers.

As can be seen from Table 4.6 on page 35, experiments using the strict Ethnologue-based data set resulted in similar results to the experiments using the modified Ethnologue data set for the two distance metrics in question. Using NJ as the tree-building algorithm, PyAline produced 13 family trees (21%) which were closer to the expert than ASJP; ASJP produced 16 family trees (26%) which were closer to the expert trees than the PyAline-produced trees for the same family, and they produced 32 equivalently close family trees (53%). Only 8 of these family trees (13%) were exactly the same, highlighting again that ASJP and PyAline are coming up with different trees most of the time. A summary of these results can be seen in Table 4.7 on page 36.

The mean difference between the RF distances to the expert trees using PyAline and ASJP is 1.87. The standard deviation of the differences between RF distances to the expert trees is 3.95. This means that 95% of the trees are within 9.76 RF distance points of each other. These statistics reinforce the previous assertion that PyAline and ASJP are about equally good at producing family trees that approximate those generated by experts, even though in most cases they produce different trees.

When using UPGMA as the tree-building algorithm, PyAline produced 16 family trees (26%) which were closer to the expert tree than those produced by ASJP. ASJP also produced 16 family

Language Family	ALINE NJ	ASJP NJ	ALINE-ASJP NJ	ALINE UPGMA	ASJP UPGMA	ALINE-ASJP UPGMA
Afro-Asiatic	348	352	150	354	350	114
Algic	46	46	28	46	48	26
Altaic	136	132	58	140	134	80
Austronesian	1424	1396	640	1434	1406	730
Arawakan	78	74	26	76	74	34
Austro-Asiatic	58	52	30	62	60	36
Australian	268	266	122	262	266	136
Border	24	26	12	26	28	14
Chukotko-Kamchatkan	0	0	0	0	0	0
Carib	28	28	12	32	30	12
Chibchan	24	26	14	24	26	18
Choco	8	8	0	8	8	2
Creole	74	74	24	74	78	36
Dravidian	22	22	8	24	24	10
Eskimo-Aleut	10	10	2	10	10	0
Eleman	4	4	2	4	4	0
Andamanese	14	14	2	16	16	0
Gogodala-Suki	10	10	4	10	10	4
Hmong-Mien	18	18	6	18	18	6
Hokan	30	28	10	32	32	2
Witotoan	4	4	0	4	4	0
Indo-European	358	368	164	362	374	204
Iroquoian	2	2	2	2	2	2
Japonic	4	4	0	4	4	2
Khoisan	22	20	6	20	22	8
Tor-Kwerba	6	6	0	6	6	4
LakesPlain	42	40	12	38	40	12
Ramu-LowerSepik	16	18	2	16	16	2
Arai-Kwomtari	8	8	4	10	10	0
Macro-Ge	36	38	18	36	38	26
Morehead and Upper Maro Rivers	18	20	4	22	22	2
Mixe-Zoque	8	8	2	10	6	6
Marind	42	42	4	46	42	6
Mayan	118	114	44	118	118	44
Muskogean	4	4	0	6	6	0
Niger-Congo	884	884	444	894	886	456
NorthCaucasian	34	34	14	34	34	16
Na-Dene	26	32	20	32	28	12
Nilo-Saharan	142	146	56	150	150	66
Oto-Manguean	80	82	38	80	82	42
Panoan	28	28	18	28	28	20
Pauwasi	6	6	0	6	6	0
Penutian	18	18	10	16	18	8
Quechuan	30	30	10	30	30	20
Sino-Tibetan	244	242	130	248	244	136
Salishan	12	12	2	12	12	0
Sepik	38	36	20	40	38	24
Siouan	6	4	2	8	4	4
Sko	8	8	2	8	10	4
East Bird's Head-Sentani	10	10	0	10	10	0
Tai-Kadai	88	90	36	88	90	50
Trans-NewGuinea	462	456	200	466	452	212
Trans-NewGuinea1	6	6	2	6	6	2
Torricelli	52	50	12	52	52	16
Totonacan	12	12	2	12	12	2
Tucanoan	18	16	4	14	18	8
Tupi	78	78	28	76	78	52
Uto-Aztecan	54	56	22	58	56	28
Uralic	26	28	4	26	28	14
West Bomberai	10	10	2	10	10	0
WestPapuan	58	56	12	58	54	22

Table 4.6: This table shows the Robinson-Foulds distance between the trees generated by PyAline and the expert trees from the strict Ethnologue classification. It also shows the RF distance between the expert trees and the ASJP generated trees, and the RF distance between the ASJP and PyAline trees. This indicates how different the automatically generated trees are from each other.

Summary of NJ Results for strict Ethnologue-Based Data			
PyAline < ASJP	13 (21%)	Mean RF Difference	1.87
ASJP < PyAline	16 (26%)	Std Dev RF Difference	3.95
PyAline == ASJP	32 (53%)		
Same Tree by Both	8 (13%)		

Table 4.7: A table describing the basic outcome of the experiment in which distance matrices created by PyAline and ASJP were turned into trees using NJ and compared to strict Ethnologue-based expert trees

Summary of UPGMA Results for Strict Ethnologue-Based Data			
PyAline < ASJP	16 (26%)	Mean RF Difference	2.33
ASJP < PyAline	16 (26%)	Std Dev RF Difference	4.32
PyAline == ASJP	29 (48%)		
Same Tree by Both	11 (18%)		

Table 4.8: A table describing the basic outcome of the experiment in which distance matrices created by PyAline and ASJP were turned into trees using UPGMA and compared to strict Ethnologue-based expert trees

trees (26%) which were closer than the family trees produced by PyAline and PyAline and ASJP produced 29 equivalently close family trees (48%). 11 of these family trees (18%) were exactly the same, slightly more than was previously seen, but still not a majority by any stretch. A summary of these results can be examined in Table 4.8 on page 36.

The mean difference between the RF distances produced by both algorithms for UPGMA-generated trees was 2.33. The standard deviation of the difference between the RF distances was 4.32. This means that for the UPGMA method, 95% of the time ASJP and PyAline generated trees which have distances from the expert tree which are within 10.97 RF distance points of each other. Once again these results confirm the notion that PyAline and ASJP are equivalently successful at approximating expert family trees, though they produce different results a large majority of the time.

The last result which the strict Ethnologue dataset reinforces is the fact that NJ is a better tree-

building algorithm for this dataset than UPGMA. For PyAline, 23 of the NJ generated family trees (38%) are closer to the expert trees than the UPGMA generated. Only 7 of the UPGMA generated family trees (11%) are closer than the NJ generated for PyAline distance matrices, and the remaining 31 family trees (51%) are equidistant from the expert trees regardless of which tree building algorithm is used. Similar results can be seen for the ASJP-generated distance matrices. 23 of the NJ generated family trees (38%) are closer to the expert trees than the UPGMA expert trees. 6 of the UPGMA-generated trees (10%) are closer to the expert trees. 32 of the UPGMA- and NJ-generated family trees (52%) are equidistant from the expert trees based on the strict Ethnologue classifications. These results reinforce those derived from the modified Ethnologue data set that NJ is a better tree generating algorithm than UPGMA for this linguistic data.

An examination of the PyAline-generated tree and ASJP-generated tree for one particular family of languages shows how close both trees are able to get to the expert trees in practice. In Figure 4.1 on page 39 we see the PyAline and ASJP trees for some Salishan languages along with the expert tree. When using NJ as the tree-building algorithm for this particular data both PyAline and ASJP end up 12 RF distance points away from the expert tree, and 2 RF distance points away from each other.

Figure 4.1 on page 39 demonstrates several interesting differences between the generated trees and the expert tree. First, the generated trees do not have any groups of more than two languages on the same level. As previously explained, this is because of the way that NJ builds a tree out of a distance matrix, finding the smallest distance at each step and conglomerating the two languages which are the smallest distance from each other. Even though NJ can't group four languages together as they are in the middle of the expert tree (see Salish Straits, Songish, Samish, and Clallam), the generated trees do group these languages together properly in related subgroups. Another interesting feature of the generated trees is how close Twana and Cowlitz are to each other even though in the expert tree they are grouped far apart from each other. Both algorithms also

manage to group the two subgroups of Cowichean-Musqueam and Spokane-Thompson together, though they place them on the tree differently than the expert does. Lastly, note that PyAline matches the expert tree in hanging Bella Coola off the main branch of the tree, while ASJP groups it together in a subgroup along with the Thompson-Spokane subgroup.

In case of Siouan, shown in Figure 4.2 on page 40, a few differences are visible between the two generated trees. The PyAline-generated tree is 6 RF distance points from the expert tree. The ASJP-generated tree beats it with an RF distance of 4 to the modified Ethnologue expert tree. The key difference between the two seems to be that the ASJP tree manages to keep Tutelo closer to the Ofo-Biloxi subgroup, while PyAline places them an extra generation apart from each other, resulting in the increase in score. Both generated trees manage to keep the Hidatsa-Crow subgroup together and both keep the Ofo-Biloxi subgroup together. Both PyAline and ASJP also keep the threesome of Lahota-Winnebago-Osage together, though clearly in a bifurcated tree, not a three childed tree. This is due, once again, to NJ being incapable of building nodes with more than 2 children.

While both the Salsihan and Siouan trees clearly invite more in depth analysis based on the linguistic characteristics of the respective families, in this thesis I'm attempting to discuss the merits of the algorithms, and point out the types of structural differences that ASJP and PyAline might produce. I leave further detailed analysis of these trees to experts in the respective language families.

After having examined these trees and the preceding data summaries, it is clear that PyAline and ASJP are comparable in terms of the trees they produce and their accuracy at generating trees similar to the expert trees for the two data sets in question. It also seems clear that the methodology used for measuring their accuracy is useful for ascertaining how good a given language distance metric is.

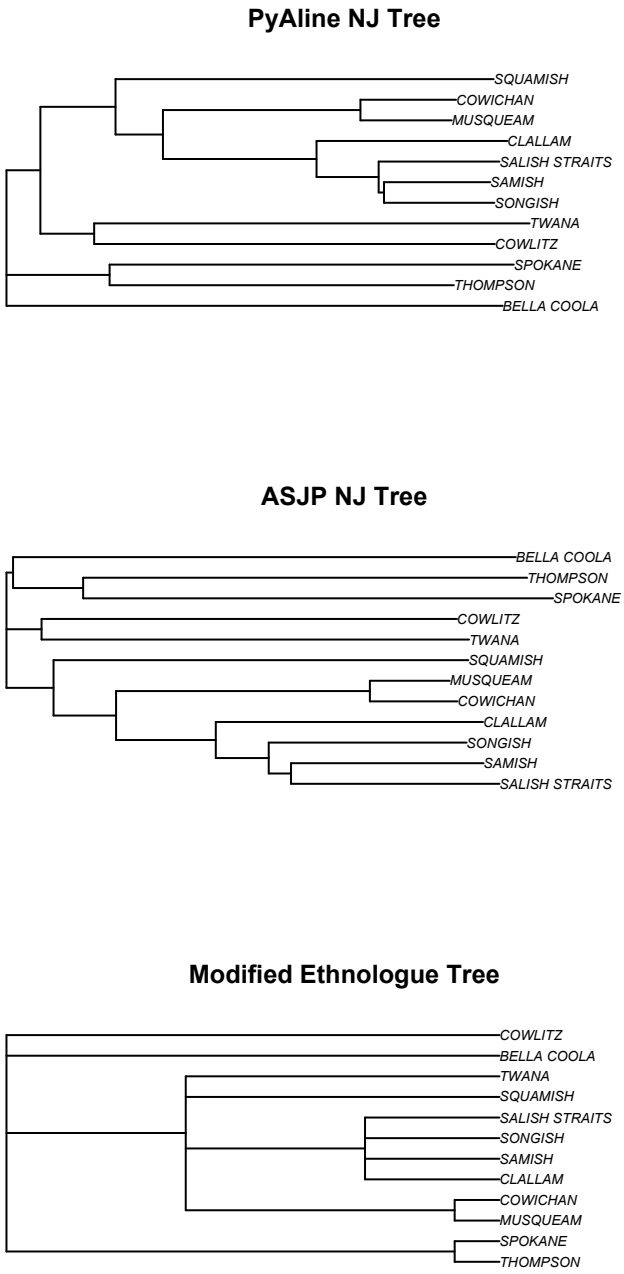


Figure 4.1: Trees generated using NJ from PyAline and ASJP distance matrices in comparison with the modified Ethnologue expert classification for the Salishan languages.

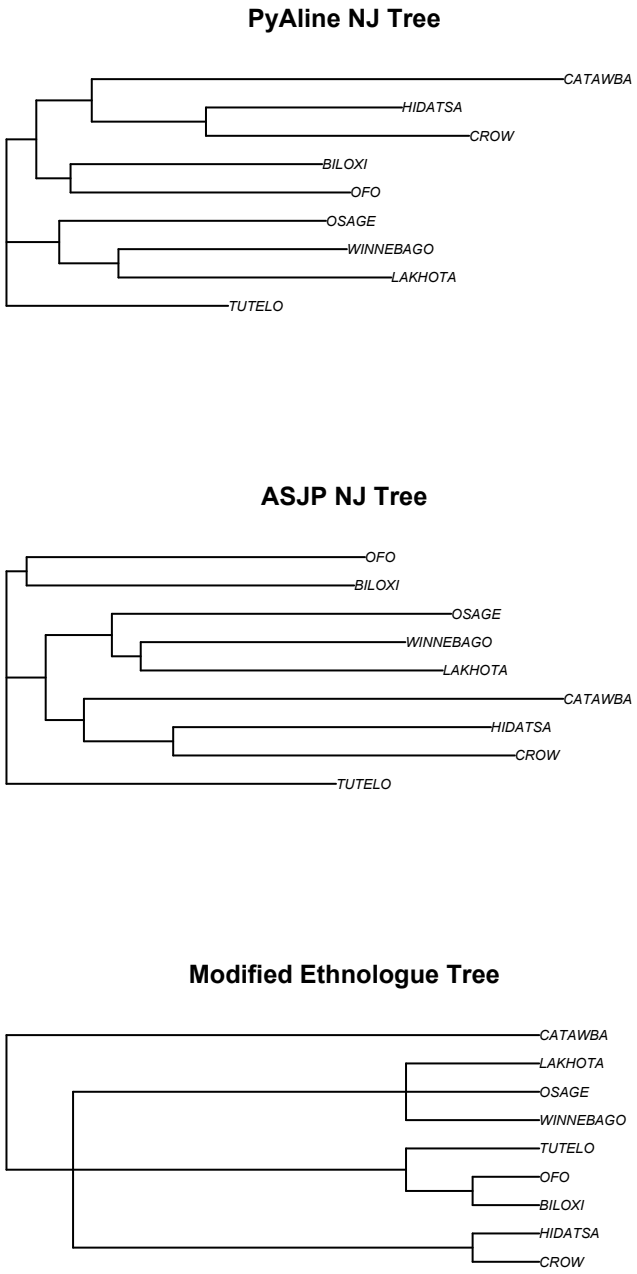


Figure 4.2: Trees generated using NJ from PyAline and ASJP distance matrices in comparison with the modified Ethnologue expert classification for the Siouan languages.

Chapter 5

Conclusion

In the preceding chapters I have proven that the ALINE distance is a viable competitor with the ASJP Project's LDND and that as far as is currently discernible with state of the art methods, the two are equally accurate at automatically generating language family trees based on Swadesh lists. Now that it has been established that the ALINE distance is a viable language distance metric, one might ask: what next? There are several avenues for further research that might be pursued armed with this knowledge. I will examine three such avenues in conclusion.

The first avenue for further research might be tuning the various parameters of the ALINE algorithm. As previously mentioned, ALINE has been designed with a variety of tunable parameters to allow the altering of the similarity scores between different phonetic strings. There are various experiments that could be conducted to determine whether tuning the parameters can yield more accurate results and if so, how to best tune them. In fact, one could easily imagine using some subset of the expert trees as a training data set to establish which values of the various parameters lead to the most accurate results for the training set and then generating trees using those parameters on the rest of the data to see if they're better everywhere. Any number of parameter training algorithms might be useful in this process, including a genetic algorithm approach or any other searching algorithm.

Another experiment in the vein of altering the various parameters of the ALINE algorithm might involve using different parameter settings for different types of language groups where some

a priori linguistic knowledge could help determine what those settings should be. For example, the weighting of the feature vectors could be set differently for families where it is known that a certain phoneme is more common, or certain phonemes are understood to be more like each other or allophones across an entire language family. Yet another experiment might involve looking at the full ASJP data set using the ALINE feature vector in order to establish those features which are most interesting mathematically using a statistical clustering method of some type. Such an algorithm could help discover mathematically which features are most useful in determining language similarity without human assitance, and then weight those features most heavily.

A second avenue for research using this data might be to come up with a database of feature-value transition probabilities. Since PyAline breaks the ASJP data down into phonetic feature vectors and is designed to actually align words, it should be relatively simple to take languages that are one step away from each other in the expert classified trees and figure out which phonemes, or which feature values, transition to one another. Using this data could generate a statistical treasure trove since it would allow linguists to definitively make assertions such as, "It is exactly N times more likely that a voiced fricative becomes unvoiced than an unvoiced fricative gains voice." This could open a whole new way of objectively measuring phonetic universals.

Having a table of phonetic transition probabilities could allow for another type of experiment using the ASJP data as well. There is another way of building trees which linguists so far linguists haven't examined, at least not on the phonetic level. Biologists have long used a type of modelling for generating family trees which differs greatly from the models that I've described. Called Monte Carlo modelling, this process uses randomization along a probablity vector to build trees based on languages. Monte Carlo modelling is akin to trying to measure the area of a design by putting it on a dart board and randomly throwing darts at it thousands of times. If you know the area of the dart board, and you know on average how many times your darts land in the shape and how many the land outside of the shape, you can determine the area of the shape by multiplying the area of

the dart board by the ratio of hits to misses on the shape.

Similarly, biologists use transition probabilities from one piece of DNA to another to simulate repeatedly the likeliness of a given tree configuration for a set of species. Since there are only 4 codons in DNA (T,C,A,G), the probabilities for one to turning into another are simple to model. Because reliable phonetic transition probabilities are as of yet unavailable, it is much harder to say, "There is an X% chance that this word in language 1 became this other word in language 2." While a phonetic transition probability table would clearly be much larger for linguistic data, it would allow this type of Monte Carlo modelling on a phonetic basis, which should theoretically lead to much more accurate trees than are currently generated.

A third avenue for future research would be transitioning from comparing automatically generated trees with expert trees to comparing automatically generated family networks with expert networks. McMahon and McMahon devote an entire chapter to using networks instead of trees in their book on automatic language classification because network structures allow more nuanced relationships between languages to be described [McMahon and McMahon, 2005]. One reason for this is that networks don't enforce a one-to-many child-to-parent relationship, which would be beneficial for a language like English which has both heavy Anglo-Saxon and Latin influences. By not imposing a strict one parent notion on English, we can create a structure which shows both of these heritages and represents how much influence each has on the language.

Networks weren't used in the current study because there is, unfortunately, no network-based representation generated by experts against which one can measure the accuracy of an algorithm for generating linguistic metrics. A further avenue of research might be to, first, create a project to gather a network based expert classification from the world's linguistic experts, and then, instead of creating family trees from the distance matrics generated by ASJP and PyAline, generate family networks and measure those against those created by experts.

Perhaps one last contribution of this thesis, however, lies not in the specific algorithm I've

explored for determining language distance, but in the methodology I've used for determining whether or not my algorithm is more accurate than ASJP's. Since the ASJP Project has made their data sets freely and publicly available for download, many other, perhaps more accurate algorithms for generating language distance matrices now have a way of determining how successful they have been at shedding more light on the origins of the worlds languages.

As linguists delve into these questions using more and more objective methodology, they will enrich our understanding of how and why our linguistic heritage has evolved as it has, continuing on in the endeavor that Calvert Watkins posed as the foundation of historical linguistics: "to demonstrate precisely how it is possible, within a linguistic tradition or traditions, for a language to pass from one system at one point in time to another system at a later point" [Watkins, 1973]. As more and larger machine readable historical data sources become available, methods such as those discussed in this thesis will nearly certainly lead the way in this type of discovery of the relationships between language systems. This is only natural as they provide better clarity and objectivity about why a linguist might argue for one type of relatedness or another. Because of this, finally statistical processes will rightfully take their place among the accepted methodologies for answering some of the earliest, and still only partially answered, questions of linguistics.

Bibliography

[Croft, 2005] Croft, W., editor (2005). *Genetic Linguistics: Essays on Theory and Method by Joseph Greenberg*. Oxford Universtiy Press, Oxford.

[Downey et al., 2008] Downey, S. S., Hallmark, B., Cox, M. P., Norquest, P., and Lansing, J. S. (2008). Computational feature-sensitive reconstruction of language relationships: Developing the aline distance for comparative historical linguistic reconstruction. *Journal of Qualitative Linguistics*, 15:340–369.

[Gascuel and Steel, 2006] Gascuel, O. and Steel, M. (2006). Neighbor-joining revealed. *Molecular Biology and Evolution*, 23:1997–2000.

[Holman et al., 2008] Holman, E. W., Wichmann, S., Brown, C. H., Velupillai, V., Müller, A., and Bakker, D. (2008). Explorations in automated language classification. *Folia Linguistica*, 42(2).

[Kondrak, 2000] Kondrak, G. (2000). A new algorithm for the alignment of phonetic sequences. *Proceedings of the First Meeting of North American Chapter of the Association for Computational Linguistics*, pages 288–295.

[Kondrak, 2002] Kondrak, G. (2002). *Algorithms for Language Reconstruction*. PhD thesis, University of Toronto, Toronto, Canada.

[Levenshtein, 1966] Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710.

[Lewis, 2009] Lewis, M. P. (2009). Ethnologue: Languages of the world.

[McMahon and McMahon, 2005] McMahon, A. M. S. and McMahon, R. (2005). *Language classification by numbers*. Oxford University Press, Oxford.

[R Development Core Team, 2010] R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

[Steel and Penny, 1993] Steel, M. A. and Penny, D. (1993). Distributions of tree comparison metrics-some new results. *Systematic Biology*, 42:126–141.

[Watkins, 1973] Watkins, C. (1973). Language and its history. *Daedalus*, 102(3):99–111.

[Wichmann et al., 2010] Wichmann, S., Holman, E. W., Bakker, D., and Brown, C. H. (2010). Evaluating linguistic distance measures. *Physica A*.