Quantifying Sound Change Regularity

Tjaden Hess
Linguistics 4424
Cornell University

Background

Traditionally, historical linguists have studied genetic relationships between languages with the help of the Comparative Method, which allows extremely accurate reconstructions of both the proto-language for the language family of interest and the phylogenetic tree expressing the genetic relationships within the language family. The Comparative Method works by first analyzing a list of presumed cognates, i.e. phonetically similar words with similar meanings in related languages. From these cognates a set of sound correspondences are constructed, e.g. German /t/ corresponds regularly with English /d/ word-initially. These correspondence classes are then presumed to be directly descended from a single proto-form, which can often be reconstructed due to the directionality of sound changes.

Recently, there has been an interest in automating this process, via automatic cognate detection and phylogenetic reconstruction, and several successful algorithms and pipelines have been proposed (**?**).

The newly emergent field of computational comparative linguistics borrows heavily from the more established domain of computational biology, adapting algorithms for gene sequence alignment and taxa clustering to the problems of phonetic sequence alignment and cognate clustering. Language, however, changes in a much more restricted manner than do nucleotide sequences; sound changes in human language are highly directional, regular and conditioned solely by phonetic environment, as opposed to the uniformly random one-off mutations that drive biological evolution. The regularity of sound change allows linguists to differentiate "true" cognate words—those derived from a single proto-form through Neogramarian sound-change—from words acquired through language contact (**?**, p. 108).

There is some definitional ambiguity in discussing sound change. A central theory of modern historical linguistics is the Neogrammarian Hypothesis, which states that *all* sound change is regular. This statement clearly is not true of every change in sound that occurs through the evolution of a language, for example the Middle English word *catched* became *caught* through analogy with words like *teach*, while *latched* did not undergo this change. The statement in fact refers specifically to so-called *Neogrammarian* sound change, which is exactly those changes which satisfy the Neogramarrian Hypothesis. Historical linguists, whenever possible, prefer to attribute language change to Neogrammarian sound change as its regularity allows the Comparative Method to function.

The question then, is to what extent is sound change Neogrammarian in nature?

Methods

The central assumption of my project is that the regularity of sound change in a language family is directly related to the ability to correctly detect cognates in a Swadesh list for that family. A Swadesh list is a general term for a wordlist chosen specifically to contain "basic" human vocabulary, i.e. vocabulary that is presumed unlikely to change through borrowing or analogy(**?**). I thus implement several recent computational methods in an attempt to quantify predictability of cognates.

Alignment

In order to even begin an investigation into computational comparative linguistics, we must at the very least have a method for aligning sequences of phonemes so that we can identify corresponding segments in potential cognates. For pairwise alignments, the Needleman-Wunsch algorithm can be used to optimally align a pair of sequences using any scoring metric (**?**). Finding optimal alignments for larger sets of words is much more difficult; while optimal algorithms exist for multiple alignment, the problem is NP-hard and so we rely on heuristic methods to give reasonable alignments. The most common technique, and the one used in my project, is *progressive alignment*, in which we align sequences sequentially along a "guide tree," which is constructed via a clustering algorithm from a distance matrix derived from pairwise alignments. While progressive alignment is not perfect, when combined with pre- and post-processing techniques it gives a very reasonable alignment in only $O(n^3)$ time (**?**).

Pairwise Alignment

In order to measure the similarity between segments, I use an adapted version of a method, first introduced by **?**, which quantifies the linguistic notion of phonetic similarity used in manual word alignment by classifying consonants into classes defined by laboratory confusion testing. More recently, a study by **?** quantified the pairwise coincidence of sound classes in manually aligned cognate sets, providing a ready-to-use similarity scoring system. **?** later extended this scoring scheme into a model call SCA, the current best performing algorithm, but for ease of implementation I used Dolgopolsky's method, which dubbed DOLGO.

To create the pairwise alignments, I use a dynamic programming technique which operates on a matrix representing optimal subsequence alignments. Each cell of the matrix

P	Labial Obstruents
T	Dental Obstruents
S	Sibilants
K	Velar Obstruents
M	Labial Nasals
N	Other Nasals
R	Liquids
W	Labial Approximant
J	Palatal Approximant
∅	Laryngials

Table 1
Dolgopolsky's Sound Classes

is filled inductively by the rule

$$M_{i,j} = \max \begin{cases} F_{i,j-1} + S(-, B_j) \\ F_{i-1,j} + S(A_i, -) \\ F_{i-1,j-1} + S(A_i, B_j) \end{cases} \tag{1}$$

where A and B are the sequences to align of length a and b respectively, S is the scoring function, and $(-)$ represents a gap (insertion or deletion). The i, jth cell of the matrix thus represents the optimal alignment of the first i characters of A with the first j characters of B.

Once the matrix is filled, we backtrack from $M_{a,b}$ to $M_{0,0}$, which gives the optimal pairwise alignment. The value of $M_{a,b}$ is the similarity score for the alignment, which will be used in calculating distance measurements for the multiple-sequence alignment.

The ability to define an arbitrary scoring function for pairs of sequences and gaps provides allows us to operate in several modes, depending largely on the penalty that we assign to insertions and deletions. In my implementation, I allow for this penalty to vary depending on whether the insert/delete operation is *opening* a new gap or *continuing* an existing gap. By manipulating this parameter, we can move along a spectrum between global and local alignment. In a global alignment, we attempt to align the full string, independent of subsequence structure while in a local alignment we care only about the best aligned subsequences and do not penalize large gaps (Fig. 1). Advanced alignment algorithms make use of a wide variety of these modes in a "library" approach, but for simplicity I use the arithmetic mean of a global mode and a local mode as my similarity metric.

It is also possible to extend the alignment algorithm so as to take into account phonetic

Seq. 1	A	B	C	X	X	X	X	X	X	A	B	C	X	X	X	Score:
Needleman	A	B	C	-	-	-	-	-	-	A	B	C	-	-	-	-3
Smith	A	B	C	-	-	-	-	-	-	A	B	C				30

Figure 1. Global vs. Local Alignment

environment, which is an important factor in measuring sound change. While I did not have time to implement this in the code base, the SCA algorithm allows for encoding conditioning prosodic environments in the tokens themselves, which can dramatically improve alignment results.

Evaluation. There are several measures of performance for sequence alignment, each with its own advantages and shortcomings (**?**); here I list the perfect alignments score (PAS), which is the percentage of alignments identical to the gold standard, the column score (CS), which reflects the percentage of correct columns in the test data, and the sum-of-pairs score (SPS), which is the proportion of all residue pairs in the gold standard that are reflected in the test data. The results of the evaluation can be seen in Table 2 compared against the SCA algorithm from the LingPy library, run on the same data. I computed the scores using LingPy's evaluation functions (**?**). The gold standard was a dataset of 16,609 words across 95 languages compiled by **?** and available at `http://dx.doi.org/10.5281/zenodo.11877`.

Model	PAS	CS	SPS
SCA	88.59	92.32	95.93
DOLGO	81.12	87.13	92.72

Table 2
Pairwise Sequence Alignments

Multiple Alignment

In order to align multiple sequences, I use the heuristic approach of progressive alignment, which traditionally proceeds in three steps:

1. A matrix is computed from pairwise distance scores are calculated for all sequences
2. A guide tree is constructed using a clustering algorithm
3. Sequences are aligned sequentially along the guide tre

Generally the implementation takes the form of a matrix-based dynamic programming algorithm, but I found it simpler to implement as a recursive function, memoized with a hash table. I also combined steps 2 and 3, merging alignments as the tree is constructed. The progressive technique demonstrates significant flexibility; at each step we merge two alignments by considering each column of the alignments to be a token, then apply the Needleman-Wunsch algorithm, parameterized by a scoring function over pairs of aligned sequences. This allows us to encode scoring functions which reward construction of columns which fall inside established sound correspondence sets.

Clustering. In order to construct the guide tree, we need a *distance* metric, while our pairwise alignment algorithm only gives pairwise similarity. The major issue with converting between the two interpretations of the measure is that the metric must be normalized in order to be directly comparable between alignments of different lengths. To adjust for this disparity, we use a technique from the ALINE algorithm by **?**.

$$d = 1 - \frac{2s}{s_1 + s_2} \tag{2}$$

where s is the raw similarity score from the pairwise alignment, and $s_1,\ s_2$ are the scores obtained from aligning the sequences with themselves. This normalizes for length and gives a distance in the range $[0, 1]$.

Given the distance metric, I used the Unweighted Pair Group Method with Arithmetic Mean (UPGMA) algorithm to construct the guide tree. UPGMA constructs a rooted tree with branch lengths from a distance matrix by iteratively joining the closest two nodes and then defining the distance to all of the other nodes as the arithmetic mean of all of the nodes in the two branches. Because UPGMA assumes ultrametricity, i.e. that all evolution happens at a constant rate across all branches, it is less accurate than newer methods like Neighbor-Joining, but its simplicity and fast running time make it a good choice for large datasets.

The algorithm works at each iteration by taking in a distance matrix and a set of rooted trees. It then joins the two nodes which have the lowest distance, producing a new node in the tree. The algorithm is defined such that by construction the distance between two clusters is exactly the mean of the distances between the elements in the cluster. Usually this is done by incrementally shrinking the distance matrix, but I found it simpler to implement a memoized *distance* function which returns the distance based on the mean-distance invariant.

Evaluation. To evaluate the performance of my multiple sequence alignment algorithm, I again used the sum-of-pairs score, and additionally the Jaccard score, which is simply the number segments in correct positions divided by the total number of segments. In multiple-alignment tests on 100 sets of phoneme sequences, my implementation slightly outperformed the SCA algorithm, although I believe that most of the discrepancy can be attributed to my handling of unknown IPA characters, which I attempted to coerce into the most similar recognized token, while the SCA implementation in LingPy simply assumes unknown IPA tokens to be vowels.

Model	SP	JC
SCA	90.11	85.03
DOLGO	92.12	87.79

Table 3
Multiple Sequence Alignment

Cognate detection

The UPGMA, besides providing a guide tree for multiple alignment also allows us to cluster words that we can hypothesize to be cognates by providing a cutoff for the maximal distance between nodes that we will join. Thus, we can feed the algorithm a list of unlabeled words and it will produce a list of possible cognate sets, along with multiple-sequence alignments for each set.

Parameters. Determining the cutoff parameter for the clustering algorithm can be difficult, as the level of phonetic similarity within cognate classes varies dramatically by language. The cutoff can be first approximated by a human, and then incrementally decreased through the iterations of the algorithm, or it can be estimated through computational methods like gradient descent. I found that first training the cutoff parameter on a small subset of the data and then using the resulting value for the rest of the analysis gave good results, i.e. the level of divergence between cognates does not change much between semantic domains, so training on a small subset of a Swadesh list is sufficient.

Language-Specific scoring

The final step in identifying cognates in wordlists is to create language-specific scoring schemes which reflect the underlying regularity of sound correspondences. In my implementation, I used an approach by **?** in which the scoring function $S_{P,Q}(x, y)$ for a residue pair x, y in alignments between languages P and Q is derived from the *attested* distribution of the residue pair in the data as contrasted with the *expected* distribution. To calculate the expected distribution, I simply count the number of times the residue pair appears for the languages in the initial multiple-alignment analysis, which provides a measure of the frequency of the residue pair among likely cognates. The expected distribution is derived by performing pairwise alignments of randomly permutations of the input data, which represents the expected distribution of the pair across unrelated words. The score is then calculated as a log odds ratio:

$$S_{P,Q}(x, y) = \frac{1}{r_1 + r_2} \left(r_1 \log_2 \left(\frac{a^2_{P,Q}(x, y)}{e^2_{P,Q}(x, y)} \right) + r_2 D(x, y) \right) \tag{3}$$

where D is the DOLGO, language agnostic, scoring function and r_1, r_2 are constants that can be adjusted to, for instance, prioritize regularity over phonetic similarity and vice versa. This equation was proposed by **?**, and arises from the intuitively super-linear growth of evidence for phoneme recurrence and the general practice of using log-odds in computational biology.

Evaluation

I tested the cognate detection component of my system on wordlists from several language families, and computed the precision, recall, and F-score against a gold standard, from **?**, using several parameters for the UPGMA cutoff. The full computation from wordlist

Dataset	Words	Concepts	Cognates
Germanic	814	110	182
Slavic	454	110	164
Japanese	1986	200	458
Romance	589	110	177

Table 4
Input data summary

Language	Precision	Recall	F-Score
Germanic	.756	.611	.6759
Slavic	.816	.969	.886
Japanese	.714	.726	.720
Romance	.768	.899	.828

Table 5
Cognate Detection Scores

to F-score takes about two minutes per language family. A summary of the test data is in Table 4 and the results can be found in Table 5

Alternative Methods

While the attested/expected distribution method works quite well in practice and is very fast compared to other methods, it does not strictly adhere to the Neogrammarian Hypothesis, a central tenant in historical linguistics which states that *sound change is regular.* the assertion is that *every* sound change is conditioned solely by its phonetic environment and occurs every time that environment occurs. While in practice this turns out to be largely definitional—irregular changes through analogy and borrowing are excluded from the term “sound change”—it has proved to be an extremely powerful tool in language reconstruction. Attempts have been made to more explicitly enumerate all of the correspondence classes present in a data set, but the major challenge to overcome in that area is the extraction of linguistically relevant details about phonetic environment of segments solely from the IPA encoded string.

? demonstrates an interesting technique using a variant of the LZ78 data compression algorithm to quantify the redundancy contained in the data, as well as several techniques to measure semantic similarity, but there is limited data to support the method’s effectiveness. **?** implemented a full Comparative Method Algorithm, but requires as input a full phylogenetic tree for the language family of interest which is often not available and requires significant human effort to construct.

In general the issue construction of explicit correspondence-set based methods is that they are far more sensitive to phonetic environment, and thus often require much more human interaction than method like the one presented here.

Conclusion

The preliminary results from my cognate detections are promising, even with rather naive methods, I was able to process large Swadesh lists quickly and identify cognates with reasonable, if varying, success. The general success indicates that at least a significant portion of sound changes are regular, although there is still far more work needed to draw any firm conclusions. With more sensitivity to phonetic context I believe that the detection rate for cognates across all language families could increase dramatically.

Note also that the test data are derived from Swadesh lists which are presumably far easier to detect true cognates in than, say in lists of technical terms or other semantic domains which would be more prone to borrowing. As a comparative measure, the cognate detection rate may well indicate something abut the depth of the phylogenetic tree, in that cognate detection is presumably worse in language families with greater time depth.

The rapid progress in computationally aided comparative linguistics in the last five years is incredibly promising, and so far the Neogrammarian hypothesis seems to be just as applicable to automated reconstruction as it is to the manual Comparative Method.

Reflection

Although the scope of my project changed slightly over its course, I am very happy with what I have implemented so far. I initially set out to detect borrowing in languages, but quickly found that the simple metric of “irregularly phonetically similar” is not sufficient to distinguish borrowings from random phonetic similarities. This is partially because the frequency of occurrence of borrowings is small in comparison to the frequency of random phonetic similarity, and partially because loanwords tend to undergo more dramatic phonetic distortion over time than I had accounted for. When I looked at the words that my alignment algorithms scored as phonetically similar, I got precision rates of under 40% and so I decided to postpone investigation in that direction. Given a few more months, I would investigate loanword distortion patterns and attempt to quantify a more stringent test for borrowings that could the be easily applied on top of my current framework.

The most significant issue that I ran into was simply the number of individual components that I needed to implement before the whole system began to come together, and the difficulties associated with maintaining a relatively large-scale project with interconnecting dependencies. Due to time constraints most of the project currently lives in one global namespace, which in a 1000+ line project can get very confusing. Given more time, my very first task would be to document and modularize the codebase. While all of the code is completely functional, it is undocumented and relies the OCaml toplevel to use; I would eventually like to package it and publish it on OPAM.

I was pleasantly surprised by the quality and availability of gold-standard data sets in this field. This is largely through the excellent work of Johann-Mattis List, the author of the excellent LingPy library and whose work formed the foundation and inspiration of this project. I also made extensive use of the evaluation module of LingPy, in order to quickly process large datasets and evaluate performance. I look forward to seeing more progress in the library in the future.

Appendix

All relevant code can by found at

`https://github.com/tjade273/compling`

All gold standard data sets are from

`http://dx.doi.org/10.5281/zenodo.11877`.