Probabilistic Models for Alignment of Etymological Data

Hannes Wettig, Roman Yangarber
Department of Computer Science
University of Helsinki, Finland
First.Last@cs.helsinki.fi

Abstract

This paper introduces several models for aligning etymological data, or for finding the best alignment at the sound or symbol level, given a set of etymological data. This will provide us a means of measuring the quality of the etymological data sets in terms of their internal consistency. Since one of our main goals is to devise automatic methods for aligning the data that are as objective as possible, the models make no a priori assumptions—e.g., no preference for vowel-vowel or consonant-consonant alignments. We present a baseline model and successive improvements, using data from Uralic language family.

1 Introduction

We present work on induction of alignment rules for etymological data in a project that studies genetic relationships among the Uralic language family. Our interest is in methods that are as objective as possible, i.e., rely only on the data rather than on prior assumptions or "universal" principles about the data, possible rules and alignments. Another goal is to derive measures of quality of data sets in terms of their internal consistency—a data-set that is more consistent should receive a higher score. We seek methods that analyze the data automatically in an unsupervised fashion. The question is whether a complete description of the correspondences can be discovered automatically, directly from raw etymological data—sets of cognate words from languages within the language family. Another way of looking at this is: what alignment rules are "inherently encoded" in a data-set (the *corpus*) itself. Thus, at present, our aim is to analyze given etymological data-sets, rather than to construct one from scratch.

Several approaches to etymological alignment have emerged over the last decade, summarized in section 2. In prior work, it was observed that etymology induction may have potential applications, among them aiding machine translation systems for resource-poor languages. Our interest is somewhat more theoretical; we are at present less interested in applications than in building models that are principled and avoid building ad-hoc heuristics into the models from the outset.

We review related work in Section 2, present a statement of the etymology alignment problem in Section 3, our models in Section 3, results in Section 5, and the next steps in Section 6.

1.1 Computational Etymology

Etymology involves several problems, including: determination of genetic relations among groups of languages, from raw linguistic data; discovering *regular sound correspondences* across languages in a given language family; reconstruction of proto-forms for a hypothetical parent language, from which the word-forms found in the daughter languages derive.

Computational etymology is interesting from the point of view of computational linguistics and machine learning. Computational methods can provide valuable feedback to the etymological/linguistic community. The methods can be evaluated by whether they perform certain aspects of etymological analysis correctly, that is, whether automatic analysis—at least in some cases—is able to produce results that match the theories established by manual analysis.

Why is computational etymology useful—can results obtained by automatic analysis clarify or improve upon established theories?

First, even if computational methods yield no new insights from the linguistic point of view, and only validate previously established theories, that would still be a useful result. Because computational approaches differ in nature from traditional linguistic methods, a matching result would serve as a non-trivial, independent confirmation of correctness of traditional methods.

Second, while some major language families have been studied extensively from the etymological perspective, many have not. Language families such as the Indo-European have received more attention than others and have been studied in greater detail, mainly because more relevant data has been collected and available to scholars for a longer time. For the less-studied language families, automatic analysis will allow linguists to bootstrap results quickly, to provide a foundation for further, more detailed investigation.

Third, the significant matter of uncertainty: Most etymological resources—dictionaries and handbooks—label certain relationships as "dubious," to a varying degree, usually due to violation of some expected regularity. Different (re)sources contain different decisions, which result in conflicts, because they are based on different theories. There is currently no way to objectively assess the relative likelihood of competing theories. Uncertainty is typically not quantified in a disciplined way, making it difficult for the linguist to know just how un/likely a particular relationship may be.

When etymology is approached by computational means, decisions are made within a rigorous framework, which makes it possible to state in probabilistic terms how likely any decision is to be correct given the data, and the relative likelihood of competing hypotheses.

Finally, a serious problem in manual etymological analysis is the potential bias of the human investigator. Bias may arise for many reasons; for example, at the time when a certain relationship is accepted as valid, some relevant data may be unknown or unavailable to the researcher, or may be available but ignored. Automatic analysis has the advantage of using all available data, without bias.

2 Related Work

We use two digital Uralic etymological resources, *SSA—Suomen Sanojen Alkuperä* ("The Origin of Finnish Words"), (Itkonen and Kulonen, 2000), and StarLing, (Starostin, 2005). StarLing was originally based on (Rédei, 1988 1991), and differs in several respects from SSA. StarLing has under 2000 Uralic cognate sets, compared with over 5000 in SSA, and does not explicitly indicate dubious etymologies. However, Uralic data in StarLing is more evenly distributed, because it is not Finnish-centric like SSA is—cognate sets in StarLing are not required to contain a member from Finnish. StarLing also gives a *reconstructed* form for each cogset, which may be useful for testing algorithms that perform reconstruction.

We are experimenting with the Uralic data by implementing algorithms modeling various etymological processes. A method due to Kondrak, (Kondrak, 2002) learns one-to-one regular sound correspondences between pairs of related languages in the data. The method in (Kondrak, 2003) finds attested complex (one-to-many) correspondences. These models are somewhat simplistic in that they operate only on one language pair at a time, and do not model the *contexts* of the sound changes, while we know that most etymological changes are conditioned on context. Our implementation of (Bouchard-Côté et al., 2007) found correspondence rules with correct contexts, using more than two languages. However, we found that this model's running time did not scale if the number of languages is above three.

In validating our experiments we use rules found in, e.g., (Lytkin, 1973; Sinor, 1997).

The Uralic language family has not been studied by computational means previously.

3 Aligning Pairs of Words

We start with pairwise alignment: aligning two languages means aligning a list of pairs of words in the two languages, which our data set claims are related. The task is *alignment*, i.e., for each pair of words, finding which symbols correspond to each other. We expect that some symbols will align with themselves, while others have gone through changes over the time that the two related languages have been evolving separately. The simplest form of such alignment at the symbol level is a pair $(s,t) \in \Sigma \times T$, a single symbol s from the *source alphabet* Σ with a symbol t from the *target alphabet* T. We denote the sizes of the alphabets by $|\Sigma|$ and $|T|$, respectively.

Clearly, this type of atomic alignment alone does not enable us to align a source word $\mathbf{s}$ of length $|\mathbf{s}|$ with a target word $\mathbf{t}$ of length $|\mathbf{t}| \neq |\mathbf{s}|$.[1] We also need to allow *insertions* and *deletions*. We augment both alphabets with the empty symbol, denoted by a dot, and write $\Sigma_.$ and $T_.$ to refer to the augmented alphabets. We can now align word pairs such as *kaikki—kõik* (meaning "all" in

[1]We use boldface to denote words, as vectors of symbols.

Finnish and Estonian), for example, as either of:

$$\begin{array}{cccccc} k & a & i & k & k & i \\ | & | & | & | & | & | \\ k & \tilde{o} & i & k & . & . \end{array} \qquad \begin{array}{ccccccc} k & a & . & i & k & k & i \\ | & | & | & | & | & | & | \\ k & . & \tilde{o} & i & k & . & . \end{array}$$

The alignment on the right consists of the pairs of symbols: (k:k), (a:.), (.:õ), (i:i), (k:k), (k:.), (i:.).

Note that we speak of "*source*" and "*target*" language for convenience only—our models are completely symmetric, as will become apparent.

3.1 The Baseline Model

We wish to encode these aligned pairs as compactly as possible, following the Minimum Description Length Principle (MDL), see e.g. (Grünwald, 2007). Given a data corpus $D = (\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_N, \mathbf{t}_N)$ of N word pairs, we first choose an alignment of each word pair $(\mathbf{s}_i, \mathbf{t}_i)$, which we then use to "transmit" the data, by simply listing the sequence of the atomic pairwise symbol alignments.[2] In order for the code to be uniquely decodable, we also need to encode the word boundaries. This can be done by transmitting a special symbol $\#$ that we do not use in any other context, only at the end of a word.

Thus, we transmit objects, or *events* e, from the event space E, in this case:

$$E = \Sigma_. \times T_. \cup \{(\# : \#)\}$$

We do this by means of Bayesian Marginal Likelihood (Kontkanen et al., 1996), or *prequential* coding, giving the total code length as:

$$\begin{aligned} L_{base}(D) = &- \sum_{e \in E} \log \Gamma\big(c(e) + \alpha(e)\big) \\ &+ \sum_{e \in E} \log \Gamma\big(\alpha(e)\big) \\ &+ \log \Gamma\left[\sum_{e \in E} \big(c(e) + \alpha(e)\big)\right] \\ &- \log \Gamma\left[\sum_{e \in E} \alpha(e)\right] \qquad (1) \end{aligned}$$

The *count* $c(e)$ is the number of times event e occurs in a complete alignment of the corpus; in particular, $c(\# : \#) = N$ occurs as many times as there are word pairs. The alignment counts are maintained in a corpus-global *alignment matrix* M, where $M(i,j) = c(i : j)$. The $\alpha(e)$ are the (Dirichlet) priors on the events. In the baseline algorithm, we set $\alpha(e) = 1$ for all e, the so-called uniform prior, which does not favour any distribution over E, *a priori*. Note that this choice nulls the second line of equation 1.

Our baseline algorithm is simple: we first randomly align the entire corpus, then re-align one word pair at a time, greedily minimizing the total cost in Eq. 1, using dynamic programming.

In the Viterbi-like matrix below in Figure 1, each cell corresponds to a partial alignment: reaching cell (i,j) means having read off i symbols of the source and j symbols of the target word. We iterate this process, *re-aligning* the word pairs; i.e., for a given word pair, we subtract the contribution of its current alignment from the global count matrix, then re-align the word pair, then add the newly aligned events back to the global count matrix. Re-alignment continues until convergence.

Re-alignment Step: align a source word σ consisting of symbols $\sigma = [\sigma_1 \ldots \sigma_n] \in \Sigma^*$ with a target word $\tau = [\tau_1 \ldots \tau_m]$. We fill in the matrix via dynamic programming, e.g., top-to-bottom, left-to-right:[3]

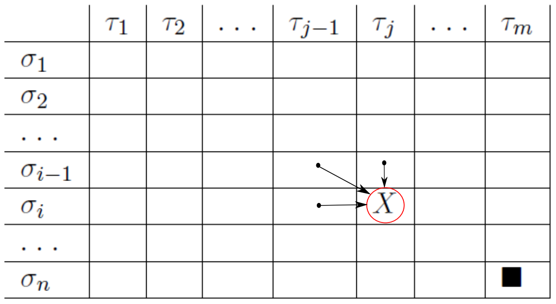

Figure 1: Dynamic programming matrix to search for most probable alignment.

Any alignment of σ and τ must correspond in a 1-1 fashion to some path through the matrix starting from top-left cell and terminating in bottom-right cell, moving only downward or rightward.

Each cell stores the probability of the *most probable* path to that point: the most probable way to have scanned the source word σ up to symbol σ_i and the target word up to τ_j, marked X in Figure 1.

[2]By *atomic* we mean that the symbols are not analyzed—in terms of their phonetic features—and treated by the baseline algorithm as atoms. In particular,

[3]NB: Figure 1 uses an extra column on the left and an extra row at the top, to store the costs for deleting symbols from the source *at the beginning of the word*, and from the target, respectively.

$$V(\sigma_i,\tau_j) = \min \begin{cases} V(\sigma_i,\tau_{j-1}) & +L(.:\tau_j) \\ V(\sigma_{i-1},\tau_j) & +L(\sigma_i:.) \\ V(\sigma_{i-1},\tau_{j-1}) & +L(\sigma_i:\tau_j) \end{cases} \quad (2)$$

In each case, the term $V(.)$ has been computed earlier by the dynamic programming; the term $L(.)$—the cost of aligning the two symbols—is a parameter of the model, computed in equation (3).

The parameters $L(e)$ or $P(e)$, for every observed event e, are computed from the *change* in the total code-length—the change that corresponds to the cost of adjoining the new event e to the set of previously observed events E:

$$\begin{aligned} L(e) &= \Delta_e L = L\big(E \cup \{e\}\big) - L(E) \\ P(e) &= 2^{-\Delta_e L} = \frac{2^{-L\left(E\cup\{e\}\right)}}{2^{-L(E)}} \end{aligned} \quad (3)$$

Combining eqs. 1 and 3 gives the probability:

$$P(e) = \frac{c(e)+1}{\sum_{e'} c(e') + |E|} \quad (4)$$

In particular, the cost of the most probable *complete* alignment of the two words will be stored in the bottom-right cell, $V(\sigma_n,\tau_m)$, marked ■.

3.2 The Two-Part Code

The baseline algorithm has revealed two problems. First, the algorithm seems to get stuck in local optima, and second, it produces many events with very low counts (occurring only once or twice).

To address the first problem we use simulated annealing with a sufficiently slow cooling schedule. This yields a reduction in the cost, and a better—more sparse—alignment count matrix.

The second problem is more substantial. Events that occur only once clearly have not got much support in the data. In theory, starting out from a common ancestor language, the number of changes that occurred in either language should be small. This does not necessarily mean many self-alignments of a symbol with itself, since a change may apply to many occurrences, e.g., all occurrences of the sound h at the end of a word have disappeared in Finnish. However, we still expect *sparse* data: we expect a relatively small portion of all *possible* events in E^+ to actually ever occur.

We incorporate this notion into our model by means of a two-part code. We first encode which events have occurred/have been observed: we first send the number of non-zero-count events—this costs $\log(|E|+1)$ bits—and then transmit which subset E^+ of the events have non-zero counts—this costs $\log\binom{|E|}{|E^+|}$ bits. This first part of the code is called the *codebook*. Given the codebook, we transmit the complete data using Bayesian marginal likelihood. The code length becomes:

$$\begin{aligned} L_{tpc}(D) = &\log(|E|+1) + \log\binom{|E|}{|E^+|} \\ &- \sum_{e\in E^+} \log\Gamma\big(c(e)+1\big) \\ &+ \log\Gamma\left[\sum_{e\in E^+}\big(c(e)+1\big)\right] - \log\Gamma(|E^+|) \end{aligned} \quad (5)$$

where E^+ denotes the set of events with non-zero counts, and we have set all $\alpha(e)$'s to one. Optimizing the above function with Simulated Annealing yields very good quality alignments.

3.3 Aligning Multiple Symbols

Multiple symbols are aligned in (Bouchard-Côté et al., 2007; Kondrak, 2003). In Estonian and Finnish appear frequent geminated consonants, which correspond to single symbols/sounds in other languages; diphthongs may align with single vowels. We allow correspondences of at most two symbols on both the source and the target side. Thus, the set of admissible kinds of events is:

$$K = \left\{ \begin{array}{lll} (\#:\#), & (\sigma:.), & (\sigma\sigma':.), \\ (.:\tau), & (\sigma:\tau), & (\sigma\sigma':t), \\ (.:\tau\tau'), & (\sigma:\tau\tau'), & (\sigma\sigma':\tau\tau') \end{array} \right\} \quad (6)$$

We do not expect correspondences of the different types to behave similarly, so we encode the occurrences of all event kinds separately in the codebook part of the two-part code:

$$L_{mult}(D) = L(CB) + L(D|CB) \quad (7)$$

$$L(CB) = \sum_{k\in K}\left[\log(N_k+1) + \log\binom{N_k}{M_k}\right] \quad (8)$$

$$\begin{aligned} L(D|CB) = &-\sum_{e\in E}\log\Gamma\big(c(e)+1\big) \\ &+ \log\Gamma\left[\sum_{e\in E}\big(c(e)+1\big)\right] - \log\Gamma(|E|) \end{aligned} \quad (9)$$

where N_k is the number of possible events of kind k and M_k the corresponding number of such events actually present in the alignment; by definition $\sum_k M_k \equiv |E|$.

Then, the parameters $P(e)$, for every observed event e, are again computed from the *change* in the code-length, eq. 3. But e may be of a kind that has been already observed previously, or it maybe of a new kind. Eq. 4 gives the formula for probability when $c(e) > 0$—that is, if $e \in E$—whereas

$$\begin{aligned} P(e) \quad = \quad & \frac{1}{\sum_{e'} c(e') + |E|} \cdot \\ & \cdot \frac{|E|}{\sum_{e'} c(e') + |E| + 1} \cdot \frac{M_k + 1}{N_k - M_k} \end{aligned} \tag{10}$$

when $e \notin E$, and e is of kind k. If the event e has been already observed, the value of $P(e)$ is computed by plugging equation (9) into eq. (3)—yielding eq. (4); if this is the first time e is observed, $P(e)$ is computed by plugging *both* eq. (9) and eq. (8) into eq. (3), since then the codebook also changes—yielding eq. (10).

Again we optimize this cost function by means of Simulated Annealing.

4 3-Dimensional Alignment

The baseline models section we restricted ourselves to aligning two languages. The alignment models allow us to learn 1-1 patterns of correspondence in the language family. The model is easily extensible to *any* number of languages. Other methods for aligning more than two languages were presented in (Bouchard-Côté et al., 2007).

We extend the 2-D model to three-dimensions as follows. We seek an alignment where symbols correspond to each other in a 1-1 fashion, as in the 2-D baseline. A three-dimensional alignment is a triplet of symbols $(\sigma : \tau : \xi) \in \Sigma \times \mathrm{T} \times \Xi$. For example, (*yhdeksän* : *üheksa* : *veχksa*)—meaning "9" in Finnish, Estonian and Mordva, can be aligned *simultaneously* as:

$$\begin{array}{ccccccccc} y & . & h & d & e & k & s & \ddot{a} & n \\ | & | & | & | & | & | & | & | & | \\ \ddot{u} & . & h & . & e & k & s & a & . \\ | & | & | & | & | & | & | & | & | \\ v & e & \chi & . & . & k & s & a & . \end{array}$$

In 3-D alignment, the input data contains all examples where words *in at least two* languages are present[4]—i.e., a word may be *missing* from one of the languages, (which allows us to utilize more of the data). Thus we have two types of examples: *complete* examples, those that have all three words present (as "9" above), and *incomplete* examples—containing words in only two languages. For example, the alignment of (*haamu*:—:*čama*)—meaning "ghost" in Finnish and Mordva—is an example where the cognate Estonian word is missing.

We must extend the 2-D alignment matrix and the 2-D Viterbi matrices to 3-D. The 3-D Viterbi matrix is directly analogous to the 2-D version. For the alignment counts in 3-D, we handle complete and incomplete examples separately.

4.1 Marginal 3-D Model

The "marginal" or "pairwise" 3-D alignment model aligns three languages simultaneously, using *only* the marginal 2-D matrices, each storing pairwise 2-D alignments. The marginal matrices for three languages are denoted $M_{\Sigma T}$, $M_{\Sigma \Xi}$ and $M_{T \Xi}$. The algorithm optimizes the total cost of the complete data, which is defined as the *sum* of the three 2-D costs obtained from applying prequential coding to the marginal alignment matrices.

When computing the cost for event $e = (\sigma, \tau, \xi)$, we consider complete and incomplete examples separately. In "incomplete" examples, we use the counts from the corresponding marginal matrix directly. E.g., for event count $c(e)$, where $e = (\sigma, -, \xi)$, and $-$ denotes the missing language, the event count is given by: $M_{\Sigma \Xi}(\sigma, \xi)$, and the cost of each alignment is computed as in the baseline model, directly in 2 dimensions.

In case when the data triplet is complete—fully observed—the alignment cost is computed as the *sum of the pairwise 2-D costs*, given by three marginal alignment count matrices:

$$\begin{aligned} L(\sigma : \tau : \xi) \quad = \quad & L_{\Sigma T}(\sigma : \tau) \\ + \quad & L_{\Sigma \Xi}(\sigma : \xi) \\ + \quad & L_{T \Xi}(\tau : \xi) \end{aligned} \tag{11}$$

The cost of each pairwise alignment is computed using prequential two-part coding, as in sec. 3.2.

Note that when we register a complete alignment (σ, τ, ξ), we register it in *each* of the base

[4]In the baseline 2-D algorithm, this requirement was also satisfied trivially, because in 2-D each example contains a word from both the source and the target language.

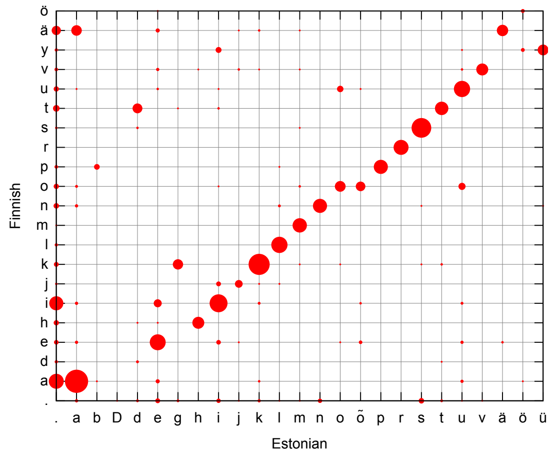

Figure 2: Alignment count matrix for Estonian-Finnish, using the two-part code.

matrices—we increment each of the marginal counts: $M_{\Sigma T}(\sigma, \tau)$, $M_{\Sigma \Xi}(\sigma, \xi)$, and $M_{T\Xi}(\tau, \xi)$. To deregister, we decrement all three counts.

To calculate the transition costs in the Viterbi algorithm, we also have two cases, complete and incomplete. For incomplete examples, we perform Viterbi in 2-D, using the costs directly from the corresponding marginal matrix, equation (5).

Note that in 3-D a non-empty symbol in one language may align to the deletion symbol "." in two languages, e.g., (.:.:d) in the 3-D example above. This means that the alignment (.:.) can now have non-zero count and marginal probability, as any other 1-1 alignment.[5]

Re-alignment: the re-alignment phase for the *complete* examples in 3-D is analogous to the re-alignment in 2-D, equation (2). The cell in the re-alignment matrix $V(\sigma_i, \tau_j, \xi_k)$—the cumulative cost of the cheapest path leading to the cell (i, j, k)—is calculated via dynamic programming, from the symbol-alignment costs $L(\sigma : \tau : \xi)$:

$$V(\sigma_i, \tau_j, \xi_k) = \min \begin{cases} V(\sigma_{i-1}, \tau_j, \xi_k) & +L(\sigma_i : . : .) \\ V(\sigma_i, \tau_{j-1}, \xi_k) & +L(. : \tau_j : .) \\ V(\sigma_i, \tau_j, \xi_{k-1}) & +L(. : . : \xi_k) \\ V(\sigma_{i-1}, \tau_{j-1}, \xi_k) & +L(\sigma_i : \tau_j : .) \\ V(\sigma_i, \tau_{j-1}, \xi_{k-1}) & +L(. : \tau_j : \xi_k) \\ V(\sigma_{i-1}, \tau_j, \xi_{k-1}) & +L(\sigma_i : . : \xi_k) \\ V(\sigma_{i-1}, \tau_{j-1}, \xi_{k-1}) & +L(\sigma_i : \tau_j : \xi_k) \end{cases}$$

[5]NB: this count is always zero in 2-D alignments, and remains impossible when aligning incomplete examples in 3-D.

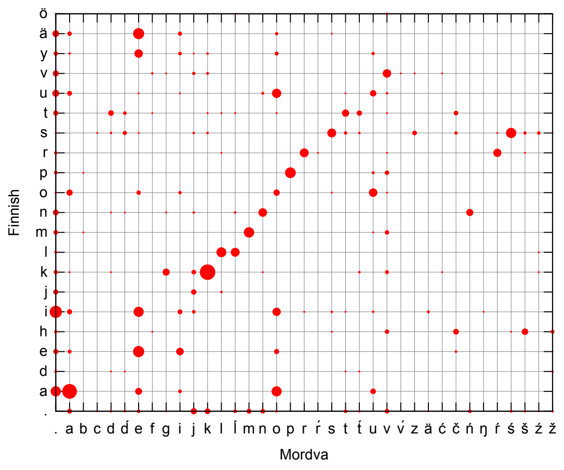

Figure 3: Mordva-Finnish 2-part code alignment.

5 Results

Evaluation of the results of the alignment algorithms is not a simple matter. One way to evaluate thoroughly would require a *gold-standard* aligned corpus; the algorithms produce alignments, which should be compared to the alignments that we would expect to find. We currently have linguists working on a gold-standard alignment for the Uralic data. Given a gold-standard alignment, we can measure performance quantitatively, e.g., in terms of accuracy.

Alignment: We can still perform qualitative evaluation, by checking how many correct sound correspondences the algorithm finds, by inspecting the final alignment of the corpus and the alignment matrix. Sample matrices for 2-D alignments of Finnish-Estonian and Finnish-Mordva (Erzä di-

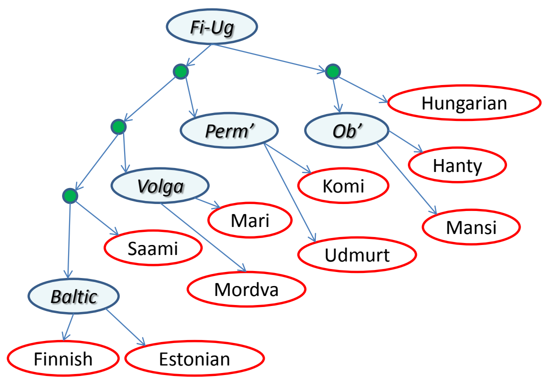

Figure 4: The Finno-Ugric sub-family of Uralic.

	est	*fin*	*khn*	*kom*	*man*	*mar*	*mrd*	*saa*	*udm*	*ugr*
est		**.372**	.702	.704	.716	.703	.665	.588	.733	.778
fin	.372		.731	.695	.754	.695	.635	.589	.699	.777
khn	.702	.719		.672	**.633**	.701	.718	.668	.712	.761
kom	.698	.703	.659		.675	.656	.678	.700	**.417**	.704
man	.702	.711	.633	.649		.676	.718	.779	.688	.752
mar	.715	.694	.731	.671	.746		**.648**	.671	.674	.738
mrd	.664	.624	.658	.678	.713	.648		.646	.709	.722
saa	.643	.589	.733	.706	.733	.621	.660		.686	.760
udm	.684	.712	.697	.417	.644	.694	.623	.677		.759
ugr	.780	.778	.761	.714	.755	.721	.743	.766	.741	

Table 1: Pairwise normalized compression costs for Finno-Ugric sub-family of Uralic, in StarLing data.

alect) are in figures 2 and 3. The size of each ball in the grid is proportional to the number of alignments in the corpus of the corresponding symbols.

Finnish and Estonian are the nearest languages in StarLing, and we observe that the alignment shows a close correspondence—the algorithm finds the *diagonal*, i.e., most sounds correspond to *"themselves"*. It must be noted that the algorithm has no *a priori* knowledge about the nature of the symbols, e.g., that Finnish *a* has any relation to Estonian *a*. The languages could be written, e.g., with different alphabets—as they are in general (we use transcribed data). This is evident in the Finnish-Estonian correspondence *y*∼*ü*, which is the same sound written using different symbols. The fact that the model finds a large number of "self" correspondences is due to the algorithm.

The model finds many Finnish-Estonian correspondences—according to rules we find in handbooks, e.g., (Lytkin, 1973; Sinor, 1997). For example, *ä*∼*a* or *ä*∼*ä* about evenly: this reflects the rule that original front vowels (as *ä*) become back in non-first syllables in Estonian. Plosives *t, k* become voiced *d, g* in certain contexts in non-initial positions. Word-final vowels *a, i, ä* are often deleted. These can be observed directly in the alignment matrix, and in the aligned corpus.

In the Finnish-Mordva alignment, the diagonal is not as pronounced, since the languages are further apart and sound correspondences more complex. Many more sounds are deleted, there is more entropy than in Finnish-Estonian; for example, many Finnish vowels map correctly to Erzä *e*, especially the front and high vowels; the back vowels do so much less often. Finnish *h* is mapped correctly to *č* or *š*. There is a (correct) preference to align *o* to *u*, and vice versa.

Compression: We can evaluate the quality of the alignment indirectly, through distances between languages. We align all languages in StarLing pairwise, using the two-part code model. We can then measure the *Normalized Compression Distance* (Cilibrasi and Vitanyi, 2005):

$$\delta(\mathbf{a},\mathbf{b}) = \frac{C(\mathbf{a},\mathbf{b}) - \min(C(\mathbf{a},\mathbf{a}),C(\mathbf{b},\mathbf{b}))}{\max(C(\mathbf{a},\mathbf{a}),C(\mathbf{b},\mathbf{b}))}$$

where $0 < \delta < 1$, and $C(\mathbf{a},\mathbf{b})$ is the compression cost—i.e., the cost of the complete aligned data for languages A and B.[6] The pairwise compression distances are shown in Table 1. Even with the simple 1x1 baseline model we see emerging patterns that mirror relationships within the Uralic family tree, shown in Fig. 4, e.g., one adapted from (Anttila, 1989). For example, scanning the row corresponding to Finnish, the compression distances *grow* as: Estonian .372, Saami .589, Mordva .635, Mari .695, Komi .695, Udmurt .699, Hanty .731, Mansi .754, and Hungarian .777, as the corresponding distance within the family tree also grows. The same holds true for Estonian.

In bold figures are sister languages, identified as being closest within their rows, (top to bottom): the Baltic, Ob', Permic, and Volgaic sub-branches.

Although the distances are not perfect (for some languages, the estimates are not 100% accurate) this confirms that the model is able to compress better—i.e., find *more regularity*—between languages that are are more closely related.

[6] $C(\mathbf{a},\mathbf{a})$ is a monolingual "alignment" of a language with itself—which is very primitive, since the 1x1 model is then able to model only the symbol frequencies.

6 Current Work and Conclusions

We have presented several models of increasing complexity for alignment of etymological datasets. The baseline 1x1 model is improved upon by introducing a two-part coding scheme and simulated annealing—this helps reduce the cost and improves the alignment. Introducing 2x2 alignment helps to reduce the cost further, but produces many spurious symbol pairs, because certain combinations of sounds appear frequently within a single language. We conclude that the proper way to handle this is by modeling context explicitly, as described above. The powerful extension of the baseline to multiple languages performs well in terms of costs and resulting alignments—these will be tested against a gold-standard in future work. An interesting consequence of the MDL-based alignment procedure, is the ability to use the alignment costs as a measure of language relation, as shown in Table 1.[7]

Although the simulated annealing heuristic already yields useful results, the algorithm still tends to end up in different final alignment states—even with a slow cooling schedule—which differ in quality in terms of the cost function, eq. 7.

We are currently extending the alignment model in two ways: by modeling context—assigning different probabilities to the same event in different environments, and by using the phonetic feature representation of the alphabet symbols.

The presented methods are not intended to replace traditional methods for etymological analysis. We are addressing only a narrow slice of the problem of etymological analysis. However, we believe these models provide an initial basis for building more interesting and complex models in the future. In particular, we can use them to approach the question of comparison of "competing" etymological data-sets or theories. The cost of an optimal alignment obtained over a given data set gives an indication of the internal regularity within the set, which can be used as an indication of consistency and quality.

We have not begun to address many important questions in etymology, including borrowing and semantics, etc. We initially focus on phonological phenomena only. Earlier work, (Kondrak, 2004) has shown that even semantics can begin to be approached in a rigorous way by computational means. Borrowing will require building models that can span across language families, which will require more mature models in the future.

[7]To save space, we focus on the Finno-Ugric sub-family of Uralic, and leave out the Samoyedic branch.

Acknowledgements

Research supported by the Uralink Project of the Academy of Finland, Grant 129185. We thank Arto Vihavainen and Suvi Hiltunen for their contribution to the implementation and testing of the algorithms. We are grateful to the anonymous reviewers for their comments and suggestions.

References

R. Anttila. 1989. *Historical and comparative linguistics*. John Benjamins.

A. Bouchard-Côté, P. Liang, T.Griffiths, and D. Klein. 2007. A probabilistic approach to diachronic phonology. In *Proc. EMNLP-CoNLL*, Prague.

R. Cilibrasi and P.M.B. Vitanyi. 2005. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4).

P. Grünwald. 2007. *The Minimum Description Length Principle*. MIT Press.

E. Itkonen and U.-M. Kulonen. 2000. *Suomen Sanojen Alkuperä (The Origin of Finnish Words)*. Suomalaisen Kirjallisuuden Seura, Helsinki, Finland.

G. Kondrak. 2002. Determining recurrent sound correspondences by inducing translation models. In *Proceedings of COLING 2002*, Taipei.

G. Kondrak. 2003. Identifying complex sound correspondences in bilingual wordlists. In *A. Gelbukh (Ed.) CICLing*, Mexico. Springer LNCS, No. 2588.

G. Kondrak. 2004. Combining evidence in cognate identification. In *Proceedings of Canadian-AI 2004*, London, ON. Springer-Verlag LNCS, No. 3060.

P. Kontkanen, P. Myllymäki, and H. Tirri. 1996. Constructing Bayesian finite mixture models by the EM algorithm. Technical Report NC-TR-97-003, ESPRIT Working Group on NeuroCOLT.

V. I. Lytkin. 1973. *Voprosy Finno-Ugorskogo Jazykoznanija (Issues in Finno-Ugric Linguistics)*, volume 1–3. Nauka, Moscow.

K. Rédei. 1988–1991. *Uralisches etymologisches Wörterbuch*. Harrassowitz, Wiesbaden.

Denis Sinor, editor. 1997. *The Uralic Languages: Description, History and Foreign Influences (Handbook of Uralic Studies)*. Brill Academic Publishers.

S. A. Starostin. 2005. Tower of babel: Etymological databases. http://newstar.rinet.ru/.