MDL-based Models for Alignment of Etymological Data

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

Abstract

We introduce several models for alignment of etymological data, that is, for finding the best alignment, given a set of etymological data, at the sound or symbol level. This is intended to obtain a means of measuring the quality of the etymological data sets, in terms of their internal consistency. 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 several successive improvements, using data from the 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. This is a continuation of previous work, reported in (Wettig and Yangarber, 2011), where the methods were introduced. In this paper, we extend the models reported earlier and give a more comprehensive evaluation of results. In addition to the attempt to induce alignment rules, we aim to derive measures of quality of data sets in terms of their internal consistency. More consistent dataset should receive a higher score in the evaluations. Currently our goal is to analyze *given*, existing etymological datasets, rather than to construct cognate sets from raw linguistic data. The question to be answered is whether a complete description of the correspondence rules can be discovered automatically. Can they be found directly from raw etymological data—sets of cognate words from languages within the language family? Are the alignment rules are "inherently encoded" in a dataset (the *corpus*) itself? We aim to develop methods that are as objective as possible, that rely only on the data, rather than on any prior assumptions about the data, the possible rules and alignments.

Computational etymology encompasses several problem areas, including: discovery of sets of genetically related words—*cognates*; determination of genetic relations among groups of languages, from raw or organized linguistic data; discovering *regular sound correspondences* across languages in a given language family; and reconstruction, either diachronic—i.e., reconstruction of proto-forms for a hypothetical parent language, from which the word-forms found in the daughter languages derive, or synchronic—i.e., of word forms that are missing from existing languages.

Several approaches to etymological alignment have emerged over the last decade. The problem of discovering cognates is addressed, e.g., in, e.g., (Bouchard-Côté et al., 2007; Kondrak, 2004; Kessler, 2001). In our work, we do not attempt to find cognate sets, but begin with given sets of etymological data for a language family, possibly different or even conflicting. We use the principle of *recurrent sound correspondence*, as in much of the literature, including the mentioned work, (Kondrak, 2002; Kondrak, 2003) and others. Modeling relationships within the language family arises in the process of evaluation of our alignment models. Phylogenetic reconstruction is studied extensively by, e.g.,(Nakhleh et al., 2005; Ringe et al., 2002; Barbancon et al., 2009); these work differ from ours in that they operate on pre-compiled sets of "characters", capturing divergent features of entire languages within the family, whereas we operate at the level of words or cognate sets. Other related work is further mentioned in the body of the paper.

We describe our datasets in the next section, present a statement of the etymology alignment problem in Section 3, cover our models in detail in Sections 4– 6, and discuss results and next steps in Section 7.

2 Data

We use two digital Uralic etymological resources, *SSA—Suomen Sanojen Alkuperä*, "The Origin of Finnish Words", (Itkonen and Kulonen, 2000), and the StarLing database, (Starostin, 2005). StarLing, originally based on (Rédei, 1988 1991), differs from SSA in several respects. StarLing has about 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. The Uralic language family has not been studied by computational means previously.

3 Aligning Pairs of Words

We begin with pairwise alignment: aligning a set of pairs of words from two related languages in our data set. The task of alignment means, for each word pair, finding which symbols correspond. We expect that some symbols will align with themselves, while others have undergone changes over the time when the two related languages have been evolving separately. The simplest form of such alignment at the symbol level is a pair $(\sigma : \tau) \in \Sigma \times T$, a single symbol σ from the *source alphabet* Σ with a symbol τ from the *target alphabet* T. We denote the sizes of the alphabets by $|\Sigma|$ and $|T|$, respectively.[1]

Clearly, with this type of 1x1 alignment alone we cannot align a source word $\boldsymbol{\sigma}$ of length $|\boldsymbol{\sigma}|$ with a target word $\boldsymbol{\tau}$ of length $|\boldsymbol{\tau}| \neq |\boldsymbol{\sigma}|$.[2] To model also *insertions* and *deletions*, we augment both alphabets with the empty symbol, denoted by a dot, and use $\Sigma_.$ and $T_.$ as augmented alphabets. We can then align word pairs such as *ien—ige*, meaning "gum" in Finnish and Established, for example, as:

$$\begin{array}{ccc} i & e & n \\ | & | & | \\ i & g & e \end{array} \qquad \begin{array}{cccc} i & . & e & n \\ | & | & | & | \\ i & g & e & . \end{array}$$

etc. The (historically correct) alignment on the right consists, e.g., of symbol pairs: (i:i), (.:g), (e:e), (n:.).

4 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; Rissanen, 1978). Given a data corpus $D = (\boldsymbol{\sigma}_1, \boldsymbol{\tau}_1), \ldots, (\boldsymbol{\sigma}_N, \boldsymbol{\tau}_N)$ of N word pairs, we first choose an alignment of each word pair $(\boldsymbol{\sigma}_i, \boldsymbol{\tau}_i)$, which we then use to "transmit" the data, by simply listing the sequence of the atomic pairwise symbol alignments.[3] 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 use only at the end of a word.

[1]We refer to "*source*" and "*target*" language for convenience only—our models are symmetric, as will become apparent.

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

[3]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, the model has no notion of identity of symbols *across* the languages!

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

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

We do this by means of Bayesian marginal likelihood, or *prequential* coding, see e.g., (Kontkanen et al., 1996), giving the total code length as:

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

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 *count 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 favor any distribution over E, *a priori*. Note that this choice nulls the second summation in 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 matrix in Fig. 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 the 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 source word $\boldsymbol{\sigma}$ consisting of symbols $\boldsymbol{\sigma} = [\sigma_1 ... \sigma_n] \in \Sigma^*$ with target word $\boldsymbol{\tau} = [\tau_1 ... \tau_m]$. We use dynamic programming to fill in the matrix, e.g., top-to-bottom, left-to-right:[4]

Alignments of $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ correspond in a 1-1 fashion to paths through the matrix, starting with cost equal to 0 in top-left cell and terminating in bottom-right cell, moving only downward or rightward.

Each cell stores the cost of the *most probable* path so far: the most probable way to have scanned $\boldsymbol{\sigma}$ up to symbol σ_i and $\boldsymbol{\tau}$ up to τ_j, marked X in the Figure:

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

Each term $V(.,.)$ has been computed earlier by the dynamic programming; the term $L(.)$—the cost of align-

[4]NB: in Fig. 1, the left column and the top row store the costs for symbol deletions *at the beginning* of the source and the target word, respectively.

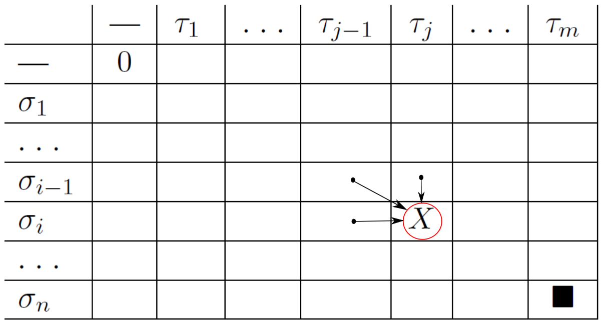

Figure 1: Re-alignment matrix: computes Dynamic Programming search for the most probable alignment.

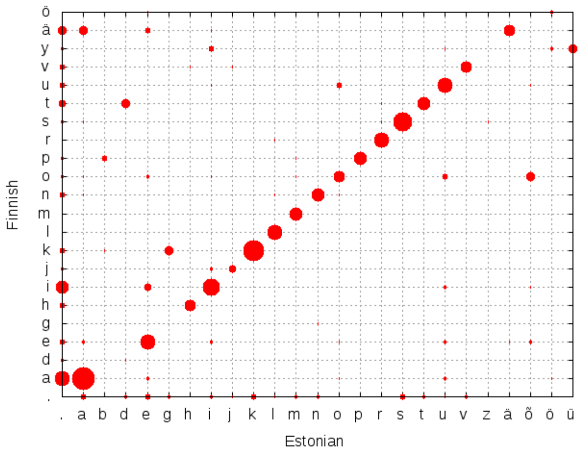

Figure 2: Global count matrix, using two-part model

ing 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:

$$L(e) \;=\; \Delta_e L = L\big(E \cup \{e\}\big) - L(E)$$

$$P(e) \;=\; 2^{-\Delta_e L} = \frac{2^{-L\left(E \cup \{e\}\right)}}{2^{-L(E)}} \qquad (3)$$

Combining eqs. 1 and 3 gives the probability:

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

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 ■. An example alignment count matrix is shown in Fig. 2.

4.1 The Two-Part Code

The baseline model revealed two problems. First, it 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. Starting from a common ancestor language, the number of changes that occurred in either language should be small. We expect *sparse* data—that only a small proportion of all *possible* events in E will actually ever occur.

We incorporate this notion into the model by means of a two-part code. First we encode which events have occurred/have been observed: we send **a.** the number of events with non-zero counts—this costs $\log(|E|+1)$ bits, and **b.** specifically which subset $E^+ \subset E$ 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, E^+, using Bayesian marginal likelihood. The code length becomes:

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

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 much better alignments.

4.2 Aligning Multiple Symbols

Multiple symbols are aligned in (Bouchard-Côté et al., 2007; Kondrak, 2003). For example, Estonian and Finnish have frequent geminated consonants, which correspond to single symbols/sounds in other languages; diphthongs may align with single vowels; etc. We extend the baseline model to a 2x2 model, to allow correspondences of up to two symbols on both the source and the target side. The set of admissible *kinds* of events is then extended to include:

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

We expect correspondences of the different types to behave differently, so we encode the occurrences of different event kinds separately in the codebook:

$$L_{mult} \;=\; L(CB) + L(Data|CB) \qquad (7)$$

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

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

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

5 Three-Dimensional Alignment

The baseline models align languages pairwise. The alignment models allow us to learn 1-1 patterns of correspondence in the language family. This model is easily extended to *any* number of languages. The model in (Bouchard-Côté et al., 2007) also aligns more than two languages at a time. 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 $(\sigma : \tau : \xi) \in \Sigma_. \times T_. \times \Xi_.$. For example, the words 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[5]—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*—where all three words present (as "9" above), and *incomplete*—containing words in only two languages. For example, for (*haamu*:—:*čama*)—"ghost" in Finnish and Mordva—the cognate Estonian word is missing.

We next extend the 2-D count matrix and the 2-D re-alignment algorithm to three dimensions. The 3-D re-alignment matrix is directly analogous to the 2-D version. For the alignment counts in 3-D, we handle complete and incomplete examples separately.

Our "marginal" 3-D alignment model aligns three languages simultaneously, using three marginal 2-D matrices, each storing a pairwise 2-D alignment. 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 word, 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 two dimensions.

[5]This was true by definition in the baseline 2-D algorithm.

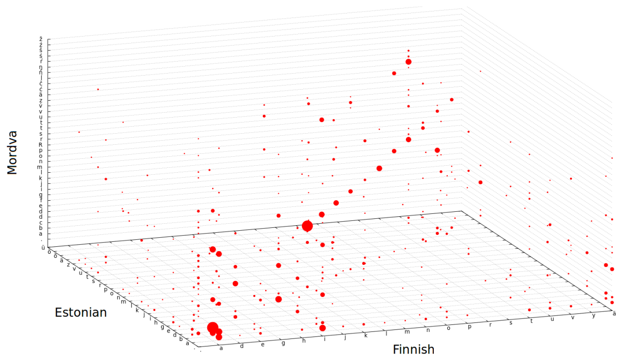

Figure 3: 3-dimensional alignment matrix.

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:[6]

$$\begin{aligned} L(\sigma : \tau : \xi) &= L_{\Sigma T}(\sigma : \tau) \\ &+ L_{\Sigma \Xi}(\sigma : \xi) \\ &+ L_{T \Xi}(\tau : \xi) \end{aligned} \tag{10}$$

The cost of each pairwise alignment is computed using prequential two-part coding, as in sec. 4.1. Note that when we register a complete alignment (σ, τ, ξ), we register it in *each* of the base 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 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).

3-D re-alignment phase: for complete examples in 3-D, is a direct analogue of the 2-D re-alignment—in the (i, j) plane—in eq. (2), extended to the third dimension, k. The cell $V(\sigma_i, \tau_j, \xi_k)$—the cost of the most probable path leading to the cell (i, j, k)—is calculated by Dynamic Programming, using the symbol-alignment costs $L(\sigma : \tau : \xi)$. In addition to the three source cells as in eq. (2), in plane k, there are four additional source cells from the previous plane, $k - 1$.

Visualization: We wish to visualize the distribution of counts in the final 3-D alignment, except that now we must deal with *expected counts*, rather than observed counts, because some of the examples are incomplete. We can form a 3-D *visualization matrix* M^* as follows:

- Compute $|D|$, the total number of alignments in the complete data (including the end-of-word alignments)

[6]Note that this results in an incomplete code, since every symbol is coded twice, but that does not affect the learning.

- For each cell (i, j, k) in M^*, the weight in that cell is given by $P(i : j : k) \cdot |D|$, where $P(i : j : k)$ is the probability of the alignment.

- The matrix of expected counts will have no zero-weight cells, since there are no zero-probability events—except $(. : . : .)$. To suppress visualizing events with very low expected counts, we don't show cells with counts below a threshold, say, 0.5.

A distribution of the expected counts in 3-D alignment is shown in figure 3. The three languages are Finnish, Estonian and Mordva. The area of each point in this figure is proportional to the expected count of the corresponding 3-way alignment.

6 Nuisance Suffixes

The existing etymological datasets are not always perfectly suited to the alignment task as we have defined it here. For example, the SSA contains mostly complete word-forms from all the languages, as they would appear in a dictionary. As a consequence, this frequently includes morphological material that is not relevant from the point of view of etymology or alignment. To illustrate this (in the Indo-European family), consider aligning English *maid* and German *mädchen*—in German, the word-form without the suffix has disappeared. Many instances with such suffixes are found in the SSA; StarLing presents *stemmed* data to a larger extent, though assuring that every form in the dataset is perfectly stemmed is a very difficult task. From the point of view of computational alignment, such "nuisance" suffixes present a problem, by confusing the model.

We extend the model to handle, or discover, the nuisance suffixes automatically, as follows. Consider, in the realignment matrix in Fig. 1, the cells (i, j) (marked X,) (i, m), and (j, n). We always end by transitioning from cell marked ■, to the terminal cell, via the special end-of-word alignment event $(\# : \#)$, whose cost is computed from N, the number of word pairs in the data (this final transition is not shown in the figure).

While previously, we could only reach the terminal cell from cell ■ via event $(\# : \#)$, we now also permit a *hyper-jump* from any cell in the matrix to the terminal cell, which is equivalent to treating the remainder of source and/or target word as a nuisance suffix. Thus, hyper-jump from cell marked X means that we code the remaining symbols $[\sigma_{i+1}...\sigma_n]$ in $\boldsymbol{\sigma}$ and $[\tau_{j+1}...\tau_m]$ in $\boldsymbol{\tau}$ separately, *not* using the global count matrix.

That is, to align $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$, we first code the symbols up to X jointly, prequentially, using the global count matrix. After X, we code a special event $(- : -)$, meaning an aligned *morpheme boundary*, similar to $(\# : \#)$ which says we have aligned the *word* boundaries. Then we code the rest of $[\sigma_{i+1}...\sigma_n]$, and the rest of $[\tau_{j+1}...\tau_m]$, both followed by $\#$.

If we hyper-jump from cell (i, m), rather than from X, then we code the event $(- : \#)$—empty suffix on target side, and then code the rest of $[\sigma_{i+1}...\sigma_n]$ in $\boldsymbol{\sigma}$ and $\#$. Symmetrically for the hyper-jump from (j, m).

	Two-part model	*Suffix model*
Fin-Est	21748.29	21445.01
Fin-Ugr	10987.98	10794.87

Table 1: Nuisance suffix models.

The cost of each symbol in the suffix can be coded, for example, according to: a uniform language model: each source symbol costs $-\log 1/(|\Sigma| + 1)$; a unigram model: for each source symbol σ (including $\#$), compute its frequency $p(\sigma)$ from the raw source data, and let $cost(\sigma) = -\log p(\sigma)$; a bigram model; etc.

Table 1 compares the code length between the original 1x1 two-part code model and a nuisance suffix model (for two language pairs). The code length is always lower in the nuisance suffix model.

Although it finds instances of true nuisance suffixes, the model may be fooled by certain phenomena. For example, when aligning Finnish and Estonian, the model decides that final vowels in Finnish which have disappeared in Estonian are suffixes, whereas that is historically not the case. To avoid such misinterpretation, the suffix detection feature should be used in conjunction with other model variants, including alignment of more than a pair of languages.

7 Results

One way to evaluate the presented models thoroughly would require a *gold-standard* aligned corpus; the models produce alignments, which would be compared to expected alignments. Given a gold-standard, we could measure performance quantitatively, e.g., in terms of accuracy. However, no gold-standard alignment for the Uralic data currently exists, and building one is very costly and slow.

Alignment: We can perform a qualitative evaluation, by checking how many correct sound correspondences a model finds—by inspecting the final alignment of the corpus and the alignment matrix. A matrix for a 2-D, 1x1 two-part model alignment of Finnish-Estonian is shown in figure 2. The size of each ball is proportional to the number of alignments in the corpus of the corresponding symbols.

Finnish and Estonian are closely related, and the alignment shows a close correspondence—the model finds the "diagonal," i.e., most sounds correspond to "themselves." We must note that this model has no *a priori* knowledge about the nature of the symbols, e.g., that Finnish *a* is identical to or has any relation to Estonian *a*. The languages are coded separately, and they may have different alphabets—as they do in general (we use transcribed data).

Rules of correspondence: One of our main goals is to model complex rules of correspondence among languages. We can evaluate the models based on how

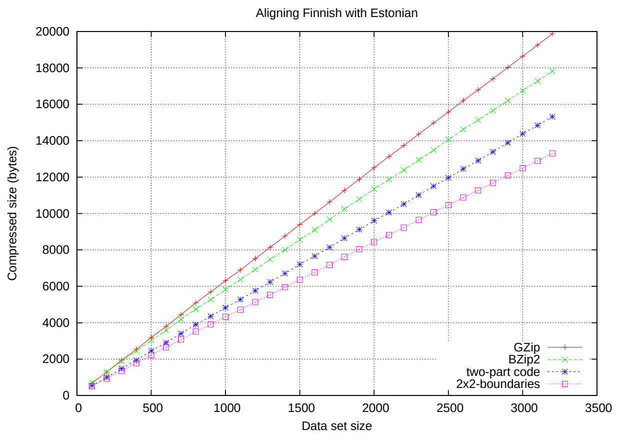

Figure 4: Comparison of compression power. Two-part code model refers to the 1x1 model that is described in section 4.1 and 2x2-boundaries model multiple symbol alignment model that is discussed in section 4.2.

well they discover rules, and how complex the rules are. In Fig. 2, the baseline model finds that Fin. *u* ∼ Est. *u*, but sometimes to *o*—this entropy is left unexplained by this model. However, the more complex 2x2 model identifies the cause exactly—by discovering that Finnish diphthongs *uo, yö, ie* correspond to Estonian long vowels *oo, öö, ee*, which covers (i.e., explains!) all instances of (*u*:*o*).

The plot shows many Finnish-Estonian correspondences, which can be found in handbooks, e.g., (Lytkin, 1973; Sinor, 1997). For example, *ä*∼*ä* vs. *ä*∼*a* about evenly—reflecting the rule that original front vowels (*ä*) became back (*a*) in non-first syllables in Estonian; word-final vowels *a*, *i*, *ä*, preserved in Finnish are often deleted in Estonian; etc. These can be observed directly in the alignment matrix, and in the aligned corpus.

Compression: In figure 4, we compare the models against standard compressors, gzip and bzip, tested on over 3200 Finnish-Estonian word pairs from SSA. The data given to our models is processed by the compressors, one word per line. Of course, our models know that they should align pairs of consecutive lines. This shows that learning about the "vertical" correspondences achieves much better compression rates—extract regularity from the data.

Language distance: We can use alignment to measure inter-language distances. We align all languages in StarLing pairwise, e.g., using a two-part 1x1 model. We can then measure the *Normalized Compression Distance* (Cilibrasi and Vitanyi, 2005):

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

where $0 < NCD < 1$, and $C(\mathbf{a}, \mathbf{b})$ is the compression cost—i.e., the cost of the complete aligned data for languages $\mathbf{a}$ and $\mathbf{b}$. The pairwise compression distances

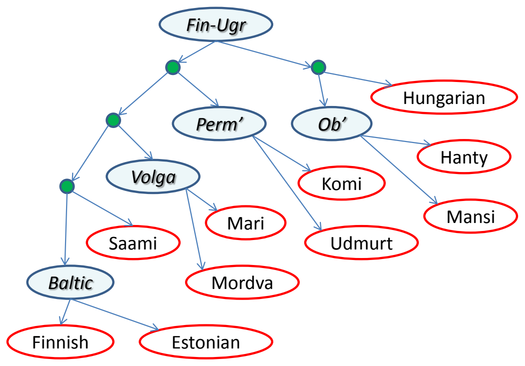

Figure 5: Finno-Ugric branch of the Uralic family

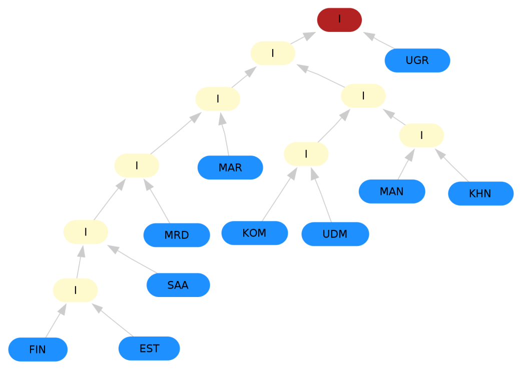

Figure 6: Finno-Ugric tree induced by NCD

are shown in table 2. We can then use these distances to draw phylogenetic trees, using hierarchical clustering methods. We used the UPGMA algorithm, (Murtagh, 1984), the resulting tree shown in Fig. 6. More sophisticated methods, such as the Fast Quartet method, CompLearn, (Cilibrasi and Vitanyi, 2011) may produce even more accurate trees. Even such a simple model as the 1x1 baseline shows emerging patterns that mirror the relationships in the Uralic family tree, shown in Fig. 5, adapted from (Anttila, 1989). For example, scanning the entries in the table corresponding to Finnish, the compression distances *grow* as the corresponding distance within the family tree grows. Sister languages (in bold) should be closest among all their relations. This confirms that the model is able to compress better—find *more regularity* in the data—between languages that are are more closely related.

8 Conclusions and Future Work

We have presented a baseline model for alignment, and several extensions. We have evaluated the models qualitatively, by examining the alignments and the rules of correspondence that they discover, and quantitatively by measuring compression cost and language distances. We trust that the methods presented here provide a good basis for further research.

We are developing methods that take *context*, or en-

	fin	*khn*	*kom*	*man*	*mar*	*mrd*	*saa*	*udm*	*ugr*
est	**.37**	.70	.70	.71	.70	.66	.58	.73	.77
fin		.73	.69	.75	.69	.63	.58	.69	.77
khn			.67	**.63**	.70	.71	.66	.71	.76
kom				.67	.65	.67	.70	**.41**	.70
man					.67	.71	.77	.68	.75
mar						**.64**	.67	.67	.73
mrd							.64	.70	.72
saa								.68	.76
udm									.75

Table 2: Pairwise normalized compression costs for Finno-Ugric sub-family of Uralic, in StarLing data.

vironment into account in modeling. The idea is to code sounds and environments as vectors of phonetic features and instead of aligning symbols, to align individual features of the symbols. The gain from introducing the context enables us to discover more complex rules of correspondence. We also plan to extend our models to diachronic reconstruction, which allows reconstruction of proto forms.

Acknowledgments

This research was supported by the Uralink Project of the Academy of Finland, grant 129185. We thank Teemu Roos for his suggestions, and Arto Vihavainen for his work on the implementation of the algorithms.

References

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

F. Barbancon, T. Warnow, D. Ringe, S. Evans, and L. Nakhleh. 2009. An experimental study comparing linguistic phylogenetic reconstruction methods. In *Proc. Conf. on Languages and Genes*, UC Santa Barbara. Cambridge University Press.

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).

R.L. Cilibrasi and P.M.B. Vitanyi. 2011. A fast quartet tree heuristic for hierarchical clustering. *Pattern Recognition*, 44(3):662–677.

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.

B. Kessler. 2001. *The Significance of Word Lists: Statistical Tests for Investigating Historical Connections Between Languages*. The University of Chicago Press, Stanford, CA.

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.

F. Murtagh. 1984. Complexities of hierarchic clustering algorithms: the state of the art. *Computational Statistics Quarterly*, 1.

L. Nakhleh, D. Ringe, and T. Warnow. 2005. Perfect phylogenetic networks: A new methodology for reconstructing the evolutionary history of natural languages. *Language*, 81(2).

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

D. Ringe, T. Warnow, and A. Taylor. 2002. Indoeuropean and computational cladistics. *Transact. Philological Society*, 100(1).

J. Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14(5).

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/.

H. Wettig and R. Yangarber. 2011. Probabilistic models for alignment of etymological data. In *Proc. NODALIDA*, Riga, Latvia.