Using context and phonetic features in models of etymological sound change

Hannes Wettig[1], **Kirill Reshetnikov**[2] and **Roman Yangarber**[1]
[1]Department of Computer Science
University of Helsinki, Finland
First.Last@cs.helsinki.fi
[2]Institute of Linguistics
Academy of Sciences
Moscow, Russia

Abstract

This paper presents a novel method for aligning etymological data, which models context-sensitive rules governing sound change, and utilizes phonetic features of the sounds. The goal is, for a given corpus of cognate sets, to find the best alignment at the sound level. We introduce an imputation procedure to compare the goodness of the resulting models, as well as the goodness of the data sets. We present evaluations to demonstrate that the new model yields improvements in performance, compared to previously reported models.

1 Introduction

This paper introduces a context-sensitive model for alignment and analysis of etymological data. Given a raw collection of etymological data (the *corpus*)—we first aim to find the “best” alignment at the sound or symbol level. We take the corpus (or possibly several different corpora) for a language family as *given*; different data sets are typically conflicting, which creates the need to determine which is more correct. Etymological data sets are found in digital etymological databases, such as ones we use for the Uralic language family. A database is typically organized into *cognate sets*; all elements within a cognate set are posited (by the database creators) to be derived from a common origin, which is a word-form in the ancestral proto-language.

Etymology encompasses several problems, including: discovery of sets of cognates—genetically related words; determination of genetic relations among groups of languages, based on linguistic data; discovering *regular sound correspondences* across languages in a given language family; and reconstruction of forms in the proto-languages.

Computational methods can provide valuable tools for the etymological community. The methods can be judged by how well they model certain aspects of etymology, and by whether the automatic analysis produces results that match theories established by manual analysis.

In this work, we allow *all* the data—and only the data—to determine what rules underly it, rather than relying on external (and possibly biased) rules that try to explain the data. This approach will provide a means of measuring the quality of the etymological data sets in terms of their internal consistency—a dataset that is more consistent should receive a higher score. We seek methods that analyze the data automatically, in an unsupervised fashion, to determine whether a complete description of the correspondences can be discovered automatically, directly from raw etymological data—cognate sets within the language family. Another way to state the question is: what alignment rules are “inherently encoded” in the given corpus itself.

At present, our aim is to analyze given etymological datasets, rather than to construct new ones from scratch. Because our main goal is to develop methods that are as objective as possible, the models make no *a priori* assumptions or “universal” principles—e.g., no preference to align vowel with vowels, or a symbol with itself. The models are not aware of the *identity* of a symbol across languages, and do not try to preserve identity, of symbols, or even of features—rather they try to find maximally regular correspondences.

In Section 2 we describe the data used in our experiments, and review approaches to etymological alignment over the last decade. We formalize the problem of alignment in Section 3, give the

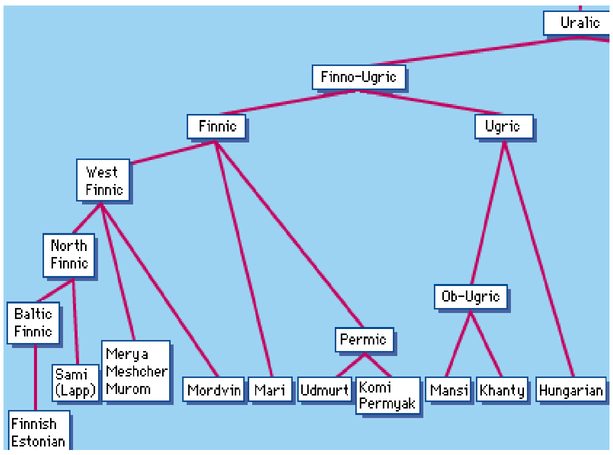

Figure 1: Finno-Ugric branch of Uralic language family (the data used in the experiments in this paper)

technical details of our models in Section 4. We present results and discussion in Sections 5 and 6.

2 Data and Related Work

We use two large Uralic etymological resources. The StarLing database of Uralic, (Starostin, 2005), based on (Rédei, 1988 1991), contains over 2500 cognate sets. *Suomen Sanojen Alkuperä* (SSA), "The Origin of Finnish Words", a Finnish etymological dictionary, (Itkonen and Kulonen, 2000), has over 5000 cognate sets, (about half of which are only in languages from the Balto-Finnic branch, closest to Finnish). Most importantly, for our models, SSA gives "dictionary" word-forms, which may contain extraneous morphological material, whereas StarLing data is mostly stemmed.

One traditional arrangement of the Uralic languages[1] is shown in Figure 1. We model etymological processes using these Uralic datasets.

The methods in (Kondrak, 2002) learn regular one-to-one sound correspondences between pairs of related languages in the data. The methods in (Kondrak, 2003; Wettig et al., 2011) find more complex (one-to-many) correspondences. These models operate on one language pair at a time; also, they do not model the *context* of the sound changes, while most etymological changes are conditioned on context. The MCMC-based model proposed in (Bouchard-Côté et al., 2007) explicitly aims to model the context of changes, and operates on more than a pair of languages.[2]

We should note that our models at present operate at the phonetic level only, they leave semantic judgements of the database creators unquestioned. While other work, e.g. (Kondrak, 2004), has attempted to approach semantics by computational means as well, our model uses the given cognate set as the fundamental unit. In our work, we do not attempt the problem of discovering cognates, addressed, e.g., in, (Bouchard-Côté et al., 2007; Kondrak, 2004; Kessler, 2001). We begin instead with a set of etymological data (or more than one set) for a language family *as given*. We focus on the principle of *recurrent sound correspondence*, as in much of the literature, including (Kondrak, 2002; Kondrak, 2003), and others.

As we develop our alignment models at the sound or symbol level, in the process of evaluation of these models, we also arrive at modeling the relationships among groups of languages within the family. Construction of phylogenies is studied extensively, e.g., by (Nakhleh et al., 2005; Ringe et al., 2002; Barbançon et al., 2009). This work differs from ours in that it operates on manually pre-selected sets of *characters*, which capture divergent features of languages within the family, whereas we operate on the raw, *complete* data.

There is extensive work on alignment in the machine-translation (MT) community, and it has been observed that methods from MT alignment may be projected onto alignment in etymology. The intuition is that translation sentences in MT correspond to cognate words in etymology, while words in MT correspond to sounds in etymology. The notion of *regularity* of sound change in etymology, which is what our models try to capture, is loosely similar to contextually conditioned correspondence of translation words across languages. For example, (Kondrak, 2002) employs MT alignment from (Melamed, 1997; Melamed, 2000); one might employ the IBM models for MT alignment, (Brown et al., 1993), or the HMM model, (Vogel et al., 1996). Of the MT-related models, (Bodrumlu et al., 2009) is similar to ours in that it is based on MDL (the Minimum Description Length Principle, introduced below).

[1]Adapted from Encyclopedia Britannica and (Anttila, 1989)

[2]Using this method, we found that the running time did not scale well for more than three languages.

3 Aligning Pairs of Words

We begin with pairwise alignment: aligning pairs of words, from two related languages in our corpus of cognates. For each word pair, the task of alignment means finding exactly which symbols correspond. Some symbols may align with "themselves" (i.e., with similar or identical sounds), while others may have undergone changes during 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 $|T|$.

To model *insertions* and *deletions*, we augment both alphabets with a special empty symbol—denoted by a dot—and write the augmented alphabets as $\Sigma_.$ and $T_.$. We can then align word pairs such as *vuosi—al* (meaning "year" in Finnish and Xanty) , for example as any of:

$$\begin{array}{ccccc} v & u & o & s & i \\ | & | & | & | & | \\ a & l & . & . & . \end{array} \quad \begin{array}{ccccc} v & u & o & s & i \\ | & | & | & | & | \\ . & a & . & l & . \end{array} \quad \text{etc...}$$

The alignment on the right then consists of the symbol pairs: (v:.), (u:a), (o:.), (s:l), (i:.).

4 Context Model with Phonetic Features

The context-aware alignment method we present here is built upon baseline models published previously, (Wettig et al., 2011), where we presented several models that do not use phonetic features or context. Similarly to the earlier ones, the current method is based on the *Minimum Description Length* (MDL) Principle, (Grünwald, 2007).

We begin with a raw set of (observed) data—the not-yet-aligned word pairs. We would like to find an alignment for the data—which we will call the *complete* data—complete with alignments, that make the most sense globally, in terms of embodying regular correspondences. We are after the regularity, and the more regularity we can find, the "better" our alignment will be (its goodness will be defined formally later). MDL tells us that the more regularity we can find in the data, the fewer bits we will need to encode it (or compress it). More regularity means lower entropy in the distribution that describes the data, and lower entropy allows us to construct a more economical code. That is, if we have no knowledge about any regularly of correspondence between symbols, the joint distribution over all possible pairs of symbols will be very flat (high entropy). If we know that certain symbol pairs align frequently, the joint distribution will have spikes, and lower entropy. In (Wettig et al., 2011) we showed how starting with a random alignment a good joint distribution can be learned using MDL. However the "rules" those baseline models were able to learn were very rudimentary, since they could not use any information in the context, and we know that many regular correspondences are conditioned by context.

We now introduce models that leverage information from the context to try to reduce the uncertainty in the distributions further, lowering the coding cost. To do that, we will code sounds in terms of their phonetic features: rather than coding the symbols (sounds) as atomic, we code them as vectors of phonetic features. Rather than aligning symbol pairs, we align the corresponding features of the symbols. While coding each feature, the model can make use of features of other sounds in its context (environment), through a special decision tree built for that feature.

4.1 Features

We will code each symbol, to be aligned in the complete data, as a feature vector. First we code the **Type** feature, with values: K (consonant), V (vowel), dot, and *word boundary*, which we denote as #. Consonants and vowels have their own sets of features, with 2–8 values per feature:

Consonant articulation		
M	**Manner**	plosive, fricative, glide, ...
P	**Place**	labial, dental, ..., velar
X	**Voiced**	– , +
S	**Secondary**	– , affricate, aspirate, ...
Vowel articulation		
V	**Vertical**	high–low
H	**Horizontal**	front–back
R	**Rounding**	– , +
L	**Length**	1–5

4.2 Contexts

While coding any symbol, the model will be allowed to query a fixed, finite set of *candidate contexts*. A context is a triplet (L, P, F), where L is the level—either source or target,—and P is one of the positions that the model may query—relative to the position currently being coded; for example, we may allow positions as in Fig. 2. F is one of the possible features found at that position. Therefore, we will have about 2 levels * 8 positions * 2–6 features $\approx$ 80 candidate contexts that can be queried by the model, as explained below.

I	itself,
–P	previous position
–S	previous non-dot symbol
–K	previous consonant
–V	previous vowel
+S	previous or self non-dot symbol
+K	previous or self consonant
+V	previous or self vowel

Figure 2: An example of a set of possible positions in the context—relative to the position currently being coded—that can be queried by the context model.

4.3 The Two-Part Code

We code the complete (i.e., aligned) data using a *two-part code*, following the MDL Principle. We first code which particular model instance we select from our *class* of models, and then code the data, given the defined model. Our model class is defined as: a set of decision trees (forest), with one tree to predict each feature on each level. The model instance will define the particular structures for each of the trees.

The forest consists of 18 decision trees, one for each feature on the source and the target level: the type feature, 4 vowel and 4 consonant features, times 2 levels. Each node in such tree will either be a leaf, or will be split by querying one of the candidate contexts defined above. The cost of coding the structure of the tree is one bit for every node—to encode whether this node was split (is an internal node) or is a leaf—plus $\approx \log 80$ times the number of internal nodes—to encode *which* particular context was chosen to split that node. We will explain how the best context to split on is chosen in Sec. 4.6.

Each feature and level define a tree, e.g., the "voiced" (**X**) feature of the source symbols corresponds to the source-**X** tree. A node N in this tree holds a distribution over the values of **X** of only those symbol instances in the complete data that have reached in N by following the context queries, starting from the root. The tree structure tells us precisely which path to follow—completely determined by the context. For example, when coding a symbol α based on another symbol found in the context of α—at some level (say, target), some position (say, –K), and one of its features (say, **M**)—the next edge down the tree is determined by that feature's value; and so on, down to a leaf. For an example of an actual decision tree learned by the model, see Fig. 5.

To compute the code length of the complete data, we only need to take into account the distributions at *the leaves*. We could choose from a variety of coding methods; the crucial point is that the chosen code will assign a particular number—the *cost*—to every possible alignment of the data. This code-length, or cost, will then serve as the *objective function*—i.e., it will be the value that the algorithm will try to optimize. Each reduction in cost will correspond directly to reduction in the entropy of the probability distribution of the symbols, which in turn corresponds to more certainty (i.e., regularity) in the correspondences among the symbols, and to improvement in the alignment. This is the link to our goal, and the reason for introducing code lengths—it gives us a single number that describes the quality of an alignment.

We use *Normalized Maximum Likelihood* (NML), (Rissanen, 1996) as our coding scheme. We choose NML because it has certain optimality properties. Using NML, we code the distribution at each leaf node separately, and summing the costs of all leaves gives the total cost of the aligned data—the value of our objective function.

Suppose n instances end up in a leaf node N, of the λ-level tree, for feature F having k values (e.g., consonants satisfying N's context constraints in the source-**X** tree, with $k = 2$ values: $-$ and $+$), and the values are distributed so that n_i instances have value i (with $i \in \{1, \ldots, k\}$). Then this requires an NML code-length of

$$\begin{aligned} L_{NML}(\lambda; F; N) &= -\log P_{NML}(\lambda; F; N) \\ &= -\log \frac{\prod_i \left(\frac{n_i}{n}\right)^{n_i}}{C(n,k)} \end{aligned} \qquad (1)$$

Here $\prod_i \left(\frac{n_i}{n}\right)^{n_i}$ is the maximum likelihood of the multinomial data at node N, and the term

$$C(n,k) = \sum_{n'_1+\ldots+n'_k=n} \prod_i \left(\frac{n'_i}{n}\right)^{n'_i} \qquad (2)$$

is a normalizing constant to make P_{NML} a probability distribution.

In the MDL literature, e.g., (Grünwald, 2007), the term $-\log C(n,k)$ is called the *stochastic complexity* or the *(minimax) regret* of the model, (in this case, the multinomial model). The NML distribution provides the unique solution to the minimax problem posed in (Shtarkov, 1987),

$$\min_{\hat{P}} \max_{\mathbf{x^n}} \log \frac{P(\mathbf{x^n}|\hat{\mathbf{\Theta}}(\mathbf{x^n}))}{\hat{P}(\mathbf{x^n})} \tag{3}$$

where $\hat{\Theta}(\mathbf{x^n}) = \arg\max_{\mathbf{\Theta}} \mathbf{P}(\mathbf{x^n})$ are the *maximum likelihood parameters* for the data $\mathbf{x^n}$. Thus, P_{NML} minimizes the worst-case regret, i.e., the number of excess bits in the code as compared to the best model in the model class, with hind-sight. For details on the computation of this code length see (Kontkanen and Myllymäki, 2007).

Learning the model from the observed data now means aligning the word pairs and building the decision trees in such a way as to minimize the two-part code length: the sum of the model's code length—to encode the structure of the trees,—and the data's code length—to encode the aligned word pairs, using these trees.

4.4 Summary of the Algorithm

The full learning algorithm runs as follows:

We start with an initial *random* alignment for each pair of words in the corpus, i.e., for each word pair choose some random path through the matrix depicted in Figure 3.

From then on we alternate between two steps: **A.** re-build the decision trees for all features on source and target levels, and **B.** re-align all word pairs in the corpus. Both of these operations monotonically decrease the two-part cost function and thus compress the data.

We continue until we reach convergence.

4.5 Re-alignment Procedure

To align source word $\vec{\sigma}$ consisting of symbols $\vec{\sigma} = [\sigma_1 ... \sigma_n]$, $\vec{\sigma} \in \Sigma^*$ with target word $\vec{\tau} = [\tau_1 ... \tau_m]$ we use dynamic programming. The tree structures are considered fixed, as are the alignments of all word pairs, except the one currently being aligned—which is subtracted from the counts stored at the leaf nodes.

We now fill the matrix V, left-to-right, top-to-bottom. Every possible alignment of $\vec{\sigma}$ and $\vec{\tau}$ corresponds to exactly one path through this matrix: starting with cost equal to 0 in the top-left cell, moving only downward or rightward, and terminating in the bottom-right cell. In this Viterbi-like matrix, every cell corresponds to a partially completed alignment: reaching cell (i,j) means having read off i symbols of the source word and j symbols of the target. Each cell $V(i,j)$—marked X in the Figure—stores the cost of the *most probable* path so far: the most probable way to have scanned $\vec{\sigma}$ through symbol σ_i and $\vec{\tau}$ through τ_j:

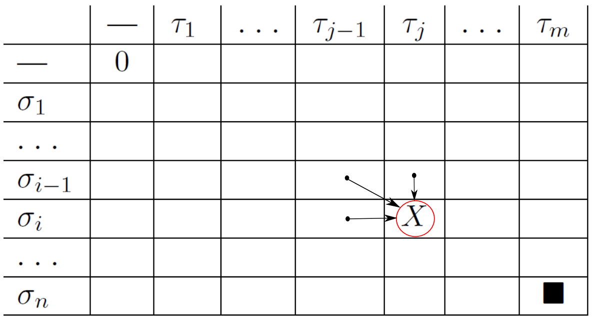

Figure 3: Dynamic programming matrix V, to search for the most probable alignment

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

Each term $V(\cdot,\cdot)$ has been computed earlier by the dynamic programming; the term $L(\cdot)$—the cost of aligning the two symbols, inserting or deleting—is determined by the change in *data* code length it induces to add this event to the corresponding leaf in all the feature trees it concerns.

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

4.6 Building Decision Trees

Given a complete alignment of the data, we need to build a decision tree, for each feature on both levels, yielding the lowest two-part cost. The term "decision tree" is meant in a probabilistic sense here: instead of a single value, at each node we store a *distribution* of the corresponding feature values, over all instances that reach this node. The distribution at a leaf is then used to code an instance when it reaches the leaf in question. We code the features in some fixed, pre-set order, and source level before target level.

We now describe in detail the process of building the tree for feature **X**, for the source level, (we will need do the same for all other features, on both levels, as well). We build this tree as follows. First, we collect all instances of consonants on the source level, and gather the the counts for feature **X**; and build an initial count vector; suppose it is:

value of **X**:	+	–
	1001	1002

This vector is stored at the *root* of the tree; the cost of this node is computed using NML, eq. 1.

Next, we try to split this node, by finding such a context that if we query the values of the feature in that context, it will help us reduce the entropy in this count vector. We check in turn all possible candidate contexts, (L, P, F), and choose the best one. Each candidate refers to some symbol found on the source (σ) or the target (τ) level, at some relative position P, and to one of that symbol's features F. We will condition the split on the possible values of F. For each candidate, we try to split on its feature's values, and collect the resulting alignment counts.

Suppose one such candidate is (σ, –V, **H**), i.e., (source-level, previous vowel, Horizontal feature), and suppose that the **H**-feature has two values: *front/back*. The vector at the root node (recall, this tree is for the **X**-feature) would then split into two vectors, e.g.:

value of **X**:	+	–
$\mathbf{X} \mid \mathbf{H}$=*front*	1000	1
$\mathbf{X} \mid \mathbf{H}$=*back*	1	1001

This would likely be a very good split, since it reduces the entropy of the distribution in each row almost to zero. The criterion that guides the choice of the best candidate to use for splitting a node is the sum of the *code lengths* of the resulting split vectors, and the code length is proportional to the entropy.

We go through all candidates exhaustively, and greedily choose the one that yields the greatest reduction in entropy, and drop in cost. We proceed recursively down the tree, trying to split nodes, and stop when the total tree cost stops decreasing.

This completes the tree for feature **X** on level σ. We build trees for all features and levels similarly, from the current alignment of the complete data.

We augment the set of possible values at every node with two additional special branches: $\neq$, meaning the symbol at the queried position is of the wrong type and does not have the queried feature, and $\#$, meaning the query ran past the beginning of the word.

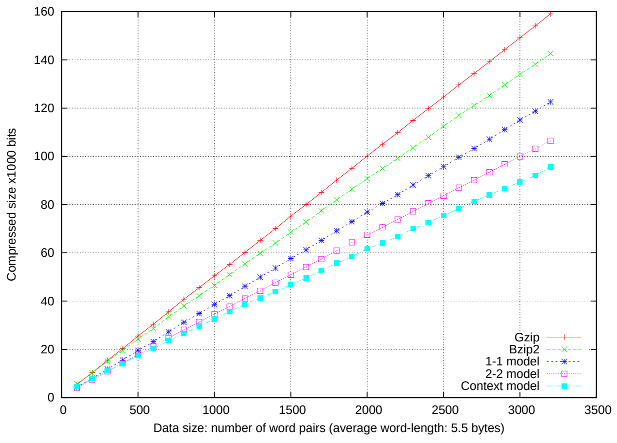

Figure 4: Comparison of compression power: Finnish-Estonian data from SSA, using the context model vs. the baseline models and standard compressors.

5 Evaluation and Results

One way to evaluate the presented models would require a *gold-standard* aligned corpus; the models produce alignments which could be compared to the gold-standard alignments, and we could measure performance quantitatively, e.g., in terms of accuracy. However, building a gold-standard aligned corpus for the Uralic data proved to be extremely difficult. In fact, it quickly becomes clear that this problem is at least as difficult as building a full reconstruction for all internal nodes in the family tree (and probably harder), since it requires full knowledge of all sound correspondences within the family. It is also compounded by the problem that the word-forms in the corpus may contain morphological material that is etymologically unrelated: some databases give "dictionary" forms, which contain extraneous affixes, and thereby obscure which parts of a given word form stand in etymological relationship with other members in the cognates set, and which do not. We therefore introduce other methods to evaluate the models.

Compression: In figure 4, we compare the context model, and use as baselines the standard data compressors, Gzip and Bzip, as well as the more basic models presented in (Wettig et al., 2011), (labeled "1x1 and "2x2"). We test the compression of up to 3200 Finnish-Estonian word pairs, from SSA. Gzip and Bzip compress data

	fin	*khn*	*kom*	*man*	*mar*	*mrd*	*saa*	*udm*	*ugr*
est	**0.26**	0.66	0.64	0.65	0.61	0.57	0.57	0.62	0.62
fin		0.63	0.64	0.65	0.59	0.56	0.50	0.62	0.63
khn			0.65	**0.58**	0.69	0.64	0.67	0.66	0.66
kom				0.63	0.68	0.66	0.70	**0.39**	0.66
man					0.68	0.65	0.72	0.62	0.62
mar						0.65	0.69	0.65	0.66
mrd							0.58	0.66	0.63
saa								0.67	0.70
udm									0.65

Table 1: Pairwise normalized edit distances for Finno-Ugric languages, on StarLing data (symmetrized by averaging over the two directions of imputation).

by finding regularities in it (i.e., frequent substrings). The comparison with Gzip is a "sanity check": we would like to confirm whether our models find more regularity in the data than would an off-the-shelf data compressor, that has no knowledge that the words in the data are etymologically related. Of course, our models know that they should align pairs of consecutive lines. This test shows that learning about the "vertical" correspondences achieves much better compression rates—allows the models to extract greater regularity from the data.

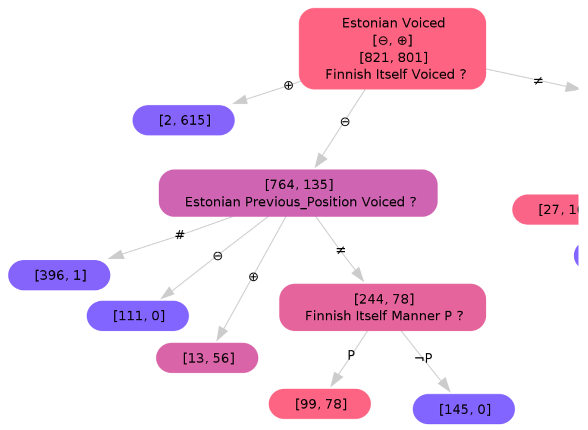

Figure 5: *Part of a tree, showing the rule for voicing of medial plosives in Estonian, conditioned on Finnish.*

Rules of correspondence: One our main goals is to model rules of correspondence among languages. We can evaluate the models based on how good they are at discovering rules. (Wettig et al., 2011) showed that aligning multiple symbols captures some of the context and thereby finds more complex rules than their 1-1 alignment model.

However, certain alignments, such as *t*∼*t/d*, *p*∼*p/b*, and *k*∼*k/g* between Finnish and Estonian, cannot be explained by the multiple-symbol model. This is due to the rule of *voicing of word-medial plosives* in Estonian. This rule could be expressed in terms of Two-level Morphology, (Koskenniemi, 1983) as: a voiceless plosive in Finnish, *may correspond* to voiced in Estonian, if not word-initial.[3] The context model finds this rule, shown in Fig. 5. This tree codes the *Target-level* (i.e., Estonian) *Voiced* consonant feature. In each node, the counts of corresponding feature values are shown in brackets. In the root node—prior to knowing anything about the environment—there is almost complete uncertainty (i.e., high entropy) about the value of *Voiced* feature of an Estonian consonant: 821 voiceless to 801 voiced in our data. Redder nodes indicate higher entropy, bluer nodes—lower entropy. The query in the root node tells us to check the context *Finnish Itself Voiced* for the most informative clue about whether the current Estonian consonant is voiced or not. Tracing the options down left to right from the root, we obtain the rules. The leftmost branch says, if the Finnish is voiced ($\oplus$), then the Estonian is almost certainly voiced as well—615 voiced to 2 voiceless in this case. If the Finnish is voiceless (Finnish Itself Voiced $= \ominus$), it says voicing *may occur*, but only in the red nodes—i.e., only if preceded by a voiced consonant on Estonian level (the branch marked by $\oplus$, 56 cases), or—if previous position is *not a consonant* (the $\neq$ branch indicates that the candidate's query does not apply: i.e., the sound found in that position is not a consonant)—it can be voiced only if the corresponding Finnish is a plosive (P, 78 cases). The blue nodes in this branch say that otherwise, the Estonian consonant almost certainly remains voiceless.

The context models discover numerous complex rules for different language pairs. For example, they learn a rule that initial Finnish *k* "changes" (corresponds) to *h* in Hungarian, if it is followed by a back vowel; the correspondence between Komi trills and Udmurt sibilants; etc.

Imputation: We introduce a novel test of the quality of the models, by using them to *impute* unseen data, as follows. For a given model, and a language pair (L_1, L_2)—e.g., (Finnish, Estonian)—hold out one word pair, and train the model on the remaining data. Then show the model the hidden Finnish word and let it guess

[3]In fact, phonetically, in modern spoken Estonian, the consonants that are written using the symbols *b,d,g* are not technically voiced, but that is a finer point, we use this rule for illustration of the principle.

the corresponding Estonian. Imputation can be done for all models with a simple dynamic programming algorithm, similar to the Viterbi-like search used during training. Formally, given the hidden Finnish string, the imputation procedure selects from all possible Estonian strings the most probable Estonian string, given the model. We then compute an edit distance between the imputed sting and the true withheld Estonian word (e.g., using the Levenshtein distance). We repeat this procedure for all word pairs in the (L_1, L_2) data set, sum the edit distances and normalize by the total size of the (true) L_2 data—this yields the Normalized Edit Distance $NED(L_2|L_1, M)$ between L_1 and L_2, under model M.

Imputation is a more intuitive measure of the model's quality than code length, with a clear practical interpretation. NED is also the ultimate test of the model's quality. If model M imputes better than M'—i.e., $NED(L_2|L_1, M) < NED(L_2|L_1, M')$—then it is difficult to argue that M could be in any sense "worse" than M'—it has learned more about the regularities between L_1 and L_2, and it knows more about L_2 given L_1. The context model, which has much lower cost than the baseline, almost always has lower NED. This also yields an important insight: it is an encouraging indication that optimizing the code length is a good approach—the algorithm does *not* optimize NED directly, and yet the cost correlates strongly with NED, which is a simple and intuitive measure of the model's quality.

6 Discussion

We have presented a novel feature-based context-aware MDL model, and a comparison of its performance against prior models for the task of alignment of etymological data. We have evaluated the models by examining the the rules of correspondence that they discovers, by comparing compression cost, imputation power and language distances induced by the imputation. The models take only the etymological data set as input, and require no further linguistic assumptions. In this regard, they is as objective as possible, given the data. The data set itself, of course, may be highly subjective and questionable.

The objectivity of models given the data now opens new possibilities for comparing entire data sets. For example, we can begin to compare the Finnish and Estonian datasets in SSA vs. StarLing, although the data sets have quite different characteristics, e.g., different size—3200 vs. 800 word pairs, respectively—and the comparison is done impartially, relying solely on the data provided. Another direct consequence of the presented methods is that they enable us to quantify uncertainty of entries in the corpus of etymological data. For example, for a given entry x in language L_1, we can compute exactly the probability that x would be imputed by any of the models, trained on all the remaining data from L_1 plus any other set of languages in the family. This can be applied equally to any entry, in particular to entries marked dubious by the database creators.

We can use this method to approach the question of comparison of "competing" etymological datasets. The cost of an optimal alignment obtained over a given data set serves as a measure of its internal consistency.

We are currently working to combine the context model with 3- and higher-dimensional models, and to extend these models to perform diachronic imputation, i.e., reconstruction of protoforms. We also intend to test the models on databases of other language families.

Acknowledgments

We are very grateful to the anonymous reviewers for their thoughtful and helpful comments. We thank Suvi Hiltunen for the implementation of the models, and Arto Vihavainen for implementing some of the earlier models. This research was supported by the Uralink Project, funded by the Academy of Finland and by the Russian Fund for the Humanities.

References

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

François G. Barbançon, Tandy Warnow, Don Ringe, Steven N. Evans, and Luay Nakhleh. 2009. An experimental study comparing linguistic phylogenetic reconstruction methods. In *Proceedings of the Conference on Languages and Genes*, UC Santa Barbara. Cambridge University Press.

Tugba Bodrumlu, Kevin Knight, and Sujith Ravi. 2009. A new objective function for word alignment. In *Proc. NAACL Workshop on Integer Linear Programming for NLP*.

Alexandre Bouchard-Côté, Percy Liang, Thomas Griffiths, and Dan Klein. 2007. A probabilistic approach to diachronic phonology. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 887–896, Prague, June.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

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

Erkki Itkonen and Ulla-Maija Kulonen. 2000. *Suomen Sanojen Alkuperä (The Origin of Finnish Words)*. Suomalaisen Kirjallisuuden Seura, Helsinki, Finland.

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

Grzegorz Kondrak. 2002. Determining recurrent sound correspondences by inducing translation models. In *Proceedings of COLING 2002: 19th International Conference on Computational Linguistics*, pages 488–494, Taipei, August.

Grzegorz Kondrak. 2003. Identifying complex sound correspondences in bilingual wordlists. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing (CICLing-2003)*, pages 432–443, Mexico City, February. Springer-Verlag Lecture Notes in Computer Science, No. 2588.

Grzegorz Kondrak. 2004. Combining evidence in cognate identification. In *Proceedings of the Seventeenth Canadian Conference on Artificial Intelligence (Canadian AI 2004)*, pages 44–59, London, Ontario, May. Lecture Notes in Computer Science 3060, Springer-Verlag.

Petri Kontkanen and Petri Myllymäki. 2007. A linear-time algorithm for computing the multinomial stochastic complexity. *Information Processing Letters*, 103(6):227–233.

Kimmo Koskenniemi. 1983. *Two-level morphology: A general computational model for word-form recognition and production*. Ph.D. thesis, University of Helsinki, Finland.

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

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

Luay Nakhleh, Don Ringe, and Tandy Warnow. 2005. Perfect phylogenetic networks: A new methodology for reconstructing the evolutionary history of natural languages. *Language (Journal of the Linguistic Society of America)*, 81(2):382–420.

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

Don Ringe, Tandy Warnow, and A. Taylor. 2002. Indo-European and computational cladistics. *Transactions of the Philological Society*, 100(1):59–129.

Jorma Rissanen. 1996. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, January.

Yuri M. Shtarkov. 1987. Universal sequential coding of single messages. *Problems of Information Transmission*, 23:3–17.

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

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of 16th Conference on Computational Linguistics (COLING 96)*, Copenhagen, Denmark, August.

Hannes Wettig, Suvi Hiltunen, and Roman Yangarber. 2011. MDL-based Models for Alignment of Etymological Data. In *Proceedings of RANLP: the 8th Conference on Recent Advances in Natural Language Processing*, Hissar, Bulgaria.