Modeling language evolution with codes that utilize context and phonetic features

Javad Nouri and **Roman Yangarber**
Department of Computer Science
University of Helsinki, Finland
first.last@cs.helsinki.fi

Abstract

We present methods for investigating processes of evolution in a language family by modeling relationships among the observed languages. The models aim to find regularities—regular correspondences in lexical data. We present an algorithm which codes the data using phonetic features of sounds, and learns long-range contextual rules that condition recurrent sound correspondences between languages. This gives us a measure of model quality: better models find more regularity in the data. We also present a procedure for imputing unseen data, which provides another method of model comparison. Our experiments demonstrate improvements in performance compared to prior work.

1 Introduction

We present work on modeling evolution within language families, by discovering regularity in data from observed languages.

The study of evolution of language families covers several problems, including: a. discovering *cognates*—"genetically related" words, i.e., words that derive from a common ancestor word in an ancestral proto-language; b. determining genetic relationships among languages in the given language family based on observed data; c. discovering patterns of sound correspondence across languages; and d. reconstruction of forms in proto-languages. In this paper, we treat a. (sets of cognates) as given, and focus on problems b. and c.[1]

Given a corpus of cognate sets,[2] we first aim to find as much regularity as possible in the data at the sound (or symbol) level.[3] An important goal is that our methods be data-driven—we aim to use all data available, and to learn the patterns of regular correspondence directly from the data. We allow only the data to determine which rules underlie it—correspondences that are *inherently encoded* in the corpus itself—rather than relying on externally supplied (and possibly biased) rules or "priors." We try to refrain from *a priori* assumptions or "universal" principles—e.g., no preference to align consonants with consonants, to align a symbol with itself, etc.

We claim that alignment may not be the best way to address the problem of regularity. Finding alignments is indeed finding a kind of regularity, but not all regularity is expressed as alignment.

The paper is organized as follows. In section 2 we review the data used in our experiments and recent approaches to modeling language evolution. We formalize the problem and present our models in section 3. The models treat sounds as vectors of phonetic *features*, and utilize the *context* of the sounds to discover patterns of regular correspondence. Once we have obtained the regularity, the question arises how we can evaluate it effectively. In section 4, we present a procedure for imputation—prediction of unseen data—to evaluate the strength of the learned rules of correspondence, by how well they predict words in one language given corresponding words in another language. We further evaluate the models by using them for building phylogenies—family trees, and comparing them to gold standards, in section 4.2. We conclude with a discussion in section 5.

We have experimented with several language families: Uralic, Turkic and Indo-European; the paper focuses on results from the Uralic family.

[1] Extending the methods to problem d. is future work.

[2] The members of a cognate set are posited (by linguists) to derive from a common, shared origin: a word-form in the (typically unobserved) ancestral proto-language.

[3] NB: we use *sounds* and *symbols* interchangeably, as we assume that input data is rendered in a phonetic transcription.

Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL), pages 136–145,
Berlin, Germany, August 7-12, 2016.

We use large-scale digital etymological resources/dictionaries. For Uralic, the StarLing database, (Starostin, 2005), contains 2586 Uralic cognate sets, based on (Rédei, 1991). The etymological dictionary *Suomen Sanojen Alkuperä* (SSA), "The Origin of Finnish Words," (Itkonen and Kulonen, 2000), has over 5000 cognate sets.

2 Related work and motivation

One traditional arrangement of the Uralic languages is shown in Figure 1; several alternative arrangements appear in the literature.

The last 15 years have seen a surge in computational modeling of language relationships, change and evolution. We provide a detailed discussion of related prior work in (Nouri et al., 2016).

In earlier work, e.g., (Wettig et al., 2011), we presented two perspectives on the problem of finding regularity. It can be seen as a problem of aligning the data. From an information-theoretic perspective, finding regularity is a problem of compression: the more regularity we find in data, the more we can compress it. In (Wettig et al., 2011), we presented baseline models, which focus on alignment of symbols, in a 1-1 fashion. We showed that aligning more than one symbol at a time—e.g., 2-2—gives better performance. Alignment is a natural way to think of comparing languages. E.g., in Figure 2, obtained by the 1-1 model, we can observe[4] that most of the time Finnish k corresponds to Estonian k (we write Fin. $k \sim$ Est. k). However, models that focus on alignments have certain shortcomings. For example, substantial probability mass is assigned to Fin. $k \sim$ Est. g, yet the model cannot explain why. Fin. $k \sim$ Est. g in certain environments—in non-first syllables, between vowels or after a voiced consonant—but the model cannot capture this regularity, because it has no notion of *context*. In fact, the regularity is much deeper: not only Fin. k, but all Finnish voiceless stops become voiced in Estonian in this environment: $p \sim b$, $t \sim d$. This type of regularity cannot be captured by the baseline model because it treats symbols as atoms, and does not know about their shared phonetic features.

We claim that alignment may always not be the best way to think about the problem of finding regularity. Figure 2 shows a prominent "diagonal,"

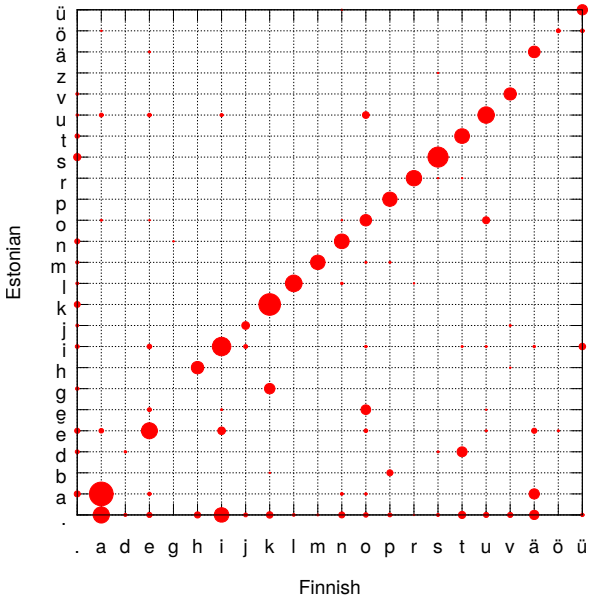

Figure 2: 1-1 alignment for Finnish and Estonian

many sounds correspond—they "align with themselves." However, as languages diverge further, this correspondence becomes blurry; e.g., when we try to align Finnish and Hungarian, the probability distribution of aligned symbols has much higher entropy, Figure 3. The reason is that the regularity lies on a much deeper level: predicting which sound occurs in a given position in a word requires knowledge of a wider context, in both Finnish and Hungarian. Hence we will prefer to think in terms of *coding*, rather than alignment.

Methods in (Kondrak, 2002), learn one-to-one sound correspondences between words in pairs of languages. Kondrak (2003), Wettig et al. (2011) find more complex—many-to-many—correspondences. These methods focus on alignment, and model *context* of the sound changes in a limited way, while it is known that most evolutionary changes are conditioned on the context of the evolving sound. Bouchard-Côté et al. (2007) use MCMC-based methods to model context, and operate on more than a pair of languages.[5]

Our models, similarly to other work, operate at the phonetic level only, leaving semantic judgements to the creators of the database. Some prior work attempts to approach semantics by computational means as well, e.g., (Kondrak, 2004; Kessler, 2001). We begin with a set of etymological data for a language family as given, and treat each cognate set as a fundamental unit of in-

[4]The size of the circle is proportional to the probability of aligning the corresponding symbols on the X and Y axes. The *dot* coordinates "." correspond to deletions/insertions.

[5]The running time did not scale well when the number of languages was above three.

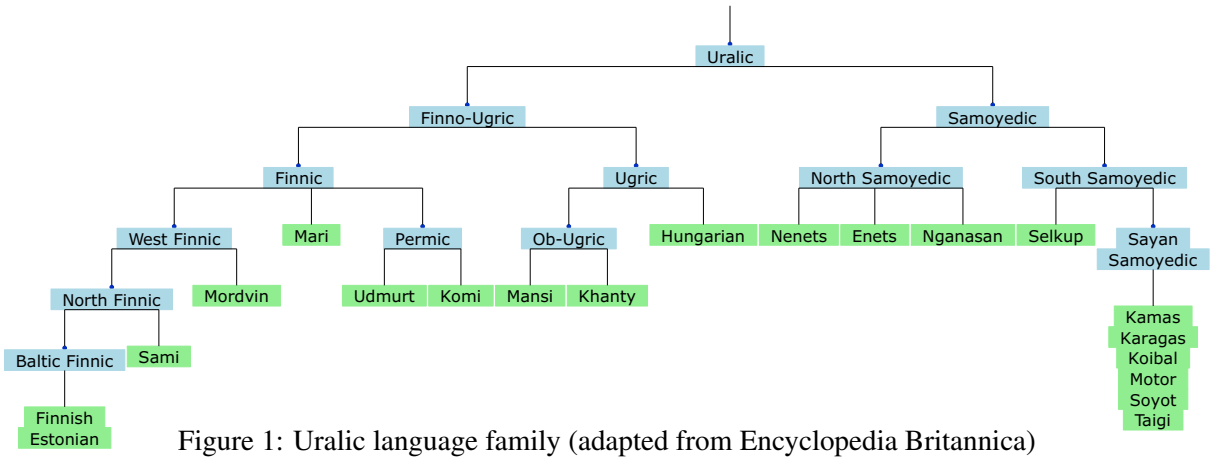

Figure 1: Uralic language family (adapted from Encyclopedia Britannica)

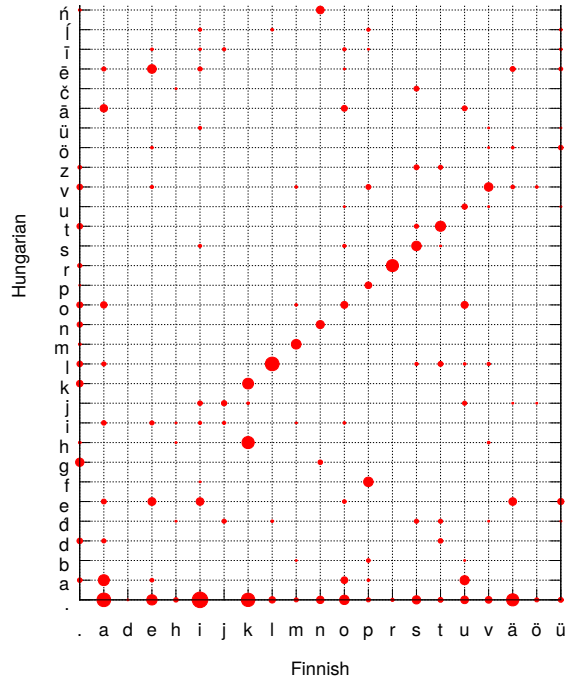

Figure 3: 1-1 alignment for Finnish and Hungarian

put. We use the principle of *recurrent sound correspondence*, as in much of the literature.

Alignment can be evaluated by measuring relationships among entire languages within the family. Construction of phylogenies is studied, e.g., in (Nakhleh et al., 2005; Ringe et al., 2002; Barbançon et al., 2009).

Our work is related to the generative "Berkeley" models, (Bouchard-Côté et al., 2007), (Hall and Klein, 2011), in the following respects.

Context: in (Wettig et al., 2011) we capture *some* context by coding *pairs* of symbols, as in (Kondrak, 2003). Berkeley models handle context by conditioning the symbol being generated upon the immediately preceding and following symbols. Our method uses broader context by building decision trees, so that non-relevant context information does not grow model complexity.

Phonetic features: in (Wettig et al., 2011) we treated sounds/symbols as atomic—not analyzed in terms of their phonetic makeup. Berkeley models use "natural classes" to define the context of a sound change, but not to generate the symbols themselves; (Bouchard-Côté et al., 2009) encode as a prior which sounds are "similar" to each other. We code symbols in terms of phonetic features. Our models are based on information-theoretic Minimum Description Length principle (MDL), e.g., (Grünwald, 2007)—unlike Berkeley. MDL brings some theoretical benefits, since models chosen in this way are guided by data with no free parameters or hand-picked "priors." The data analyst chooses the model class and structure, and the coding scheme, i.e., a decodable way to encode model and data. This determines the learning strategy—we optimize the cost function, which is the code length determined by these choices.

Objective function: we use NML—the normalized maximum likelihood, not reported previously in this setting. It is preferable for theoretical and practical reasons, e.g., to prequential coding used in (Wettig et al., 2011), as explained in section 3.1.

Models that utilize more than the immediate adjacent environment of a sound to build a complete alignment of a language family have not been reported previously, to the best of our knowledge.

3 Coding pairs of words

We begin with baseline algorithms for pairwise coding: in (Wettig et al., 2011; Wettig et al., 2012) we code pairs of words, from two related languages in our corpus of cognates. For each word pair, the task of alignment is finding which sym-

bols correspond best; the task of coding is achieving more compression. The simplest form of symbol alignment 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.

To model *insertions* and *deletions*, we augment both alphabets with a special "empty" symbol—denoted by a dot—and write the augmented alphabets as $\Sigma^{\cdot}$ and $T^{\cdot}$. We can then align word pairs, such as *hiiri—lögkər* (meaning "mouse" in Finnish and Khanty) in many different ways; putting Finnish (source level, above) and Khanty (target level, below), for example:

h	*i*	.	.	*i*	*r*	*i*		.	*h*	.	.	*i*	*i*	*r*	*i*	
|	|	|	|	|	|	|		|	|	|	|	|	|	|	|	···
l	*ö*	*ŋ*	*k*	*ə*	*r*	.		*l*	*ö*	*ŋ*	*k*	*ə*	*r*	.	.	

A final note about alignments: we find no satisfactory way to *evaluate* alignments. Which of the above alignments is "better"? It may be satisfying to prefer the left one, observing that Fin. *h* corresponds well to Khn. *l* (since they both go back to Proto-Uralic š); Fin. *r* ∼ Khn. *r*, etc. However, if a model achieves better compression by preferring the alignment on the right, then it is difficult to argue that that alignment is "not correct."

3.1 Context model with phonetic features

Our coding method is based on MDL. The most refined form of MDL, NML—Normalized Maximum Likelihood, (Rissanen, 1996)—cannot be efficiently computed for our model. Therefore, we resort to a classic two-part coding scheme. The first part of the two-part code is responsible for splitting the data into subsets corresponding to certain contexts. However, given the contexts, we *can* use NML to encode these subsets.[6]

We begin with a raw set of observed data—word pairs in two languages. We search for a way to code the data, by capturing regular correspondences. The goodness of the code is defined formally below. MDL says that the more regularity we can find in the data, the fewer bits we will need to encode (or compress) it. More regularity means lower entropy in the distribution that describes the data, and lower entropy lets us construct a more economical code.

Features: Rather than coding symbols (sounds) as atomic, we code them in terms of their phonetic features. For example, figure 4 shows how a model might code Finnish *jalka* and Estonian *jalg* (meaning "leg"). We code the symbols in a fixed order: top to bottom, left to right. Each symbol is coded as a vector of its phonetic features, e.g., $k = [\xi\ \chi\ \phi\ \psi]$.

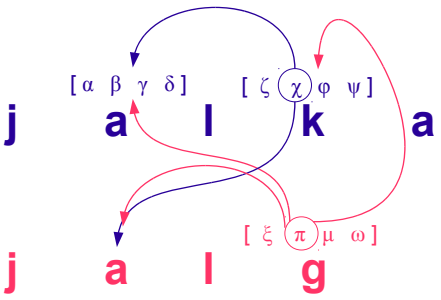

Figure 4: Fin. *jalka* (source) ∼ Est. *jalg* (target)

For each symbol, first we code a special **Type** feature, with values: K (consonant), V (vowel), dot (insertion / deletion), or # (*word boundary*).[7] Consonants and vowels have different sets of features; each feature has 2–8 values, listed in Figure 5A. Features are coded in a fixed order.

Contexts: While coding each *feature* of the symbol, the model is allowed to query a fixed and finite a set of *candidate contexts*. The idea is that the model can query its "history"—information that has already been coded previously. When coding k, e.g., the model may query features of blue a (β, γ, etc.), as well as features of red a, etc. When coding g the model may query those, and in addition also the features of k (χ, ϕ, etc.)

Formally, a context is a triplet (L, P, F): L is the level—source (σ) or target (τ); P is one of the positions that the model may query—relative to the position currently being coded; for example, we may query positions shown in Figure 5B. F is one of the possible features found at that position. Thus, we have in total about 2 levels × 8 positions × 5 features ≈ 80 candidate contexts that can be queried, as explained in detail below.

3.2 Two-part code

We code the complete (i.e., aligned) data using a two-part code, following MDL. We first code which model instance we select from our model *class*, and then code the data given the model. Our model class is defined as follows: a set of decision trees (forest)—one tree per feature per level (*separately* for source and for target). A model *instance*

[6] Theoretical benefits of NML over other coding schemes include freedom from priors, invariance to re-parametrization, and other optimality properties, which are outside the scope of this paper, (Rissanen, 1996).

[7] Type feature and word end (#) not shown in figure.

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

Contexts	
I	itself, possibly dot
–P	previous position, possibly dot
–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
...	*(other contexts possible)*

Figure 5: (A: left) Phonetic features and (B: right) phonetic contexts / environments.

will define a particular structure for each tree.

Cost of coding the structure: Thus, 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 a tree will either be a leaf, or will be split—by querying one of the candidate contexts defined above. The cost of a tree is one bit for every node n_i—to encode whether n_i is internal (was split) or a leaf—plus the number of internal nodes $\times \approx \log 80$—to encode *which* particular context was chosen to split each n_i. We explain how the model chooses the best candidate context on which to split a node in section 3.3.

Each feature and level define a tree, e.g., the "voiced" (**X**) feature of the source symbols—corresponds to the σ-**X** tree. A node N in this tree holds a distribution over the values of feature **X** of only those symbol instances in the complete data that have reached node N, by following the context queries from the root downward. The tree structure tells us precisely which path to follow—completely determined by the context. When coding a symbol α based on another symbol found in the context C of α—for example, $C = (\tau, -K, \mathbf{M})$: at level τ, position –K, and one of the features **M**—the next edge down the tree is determined by that feature's value; and so on, down to a leaf.[8]

Cost of the data given the model: is computed by taking into account only the distributions at *the leaves*. The code will assign a cost (code-length) to every possible alignment of the data. The total code-length is the *objective* function that the learning algorithm will optimize.

Coding scheme: we use Normalized Maximum Likelihood (NML), and prequential coding as in (Wettig et al., 2011). We code the distribution at each leaf node separately; the sum of the costs of all leaves gives the total cost of the complete data—the value of the objective function.

Suppose n instances reach a leaf node N, of the tree for feature F on level λ, and F has k values: e.g., n consonants satisfying N's context constraints in the σ-**X** tree, with $k = 2$ values:$\{-, +\}$. Suppose also that 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:

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

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

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

is a normalizing constant to make P_{NML} a probability distribution. In MDL literature, (Grünwald, 2007), the term $-\log C(n,k)$ is called the *parametric complexity* or the *(minimax) regret* of the model—in this case, the multinomial model.

The NML distribution is the unique solution to the mini-max problem posed in (Shtarkov, 1987),

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

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. Details on the computation of this code length are given in (Kontkanen and Myllymäki, 2007).

Learning the model from the observed data now means aligning word pairs and building decision

[8] Model code to construct trees from data, and examples of decision trees learned by the model are made publicly available on the Project Web site: *etymon.cs.helsinki.fi/*.

trees so as to minimize the two-part code length: the sum of the model's code length—encoding the structure of the trees,—and the code length of the data given the model—encoding the aligned word pairs using these trees.

Summary of the algorithm: We start with an initial *random* alignment for each pair of words in the corpus. We then 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, using dynamic programming. Both of these operations monotonically decrease the two-part cost function and thus compress the data. We continue until we reach convergence.

Simulated annealing with a slow cooling schedule is used to avoid getting trapped in local optima.

3.3 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: at each node we store a *distribution* over the respective feature values, for all instances that reach this node. The distribution at a given leaf is then used to code an instance when it reaches the leaf. We code the features in a fixed, pre-set order, and source level (σ-level) before target (τ-level).

We now describe in detail the process of building the tree—using as example a tree for the σ-level feature **X**. (We will need do the same for all other features, on both levels, as well.) First, we collect all instances of consonants on σ-level, 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. Note that this vector / distribution has rather high entropy.

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 σ-level or 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., (σ-level, previous vowel, Horizontal feature), and suppose that the **H**-feature has two values: *front / back*. Suppose also that the vector at the root node (recall, this tree is for the **X**-feature) would then split into two vectors, for example:

value of **X** →	+	–
X | **H**=*front*	1000	1
X | **H**=*back*	1	1001

This would likely be a very good split, since it reduces the entropy of the distribution in each row to near zero. The criterion that guides the choice of the best candidate context 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,[9] 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 all remaining trees—for all features and all levels similarly—based on the current alignment of the complete data.

3.4 Variations of context-based models

The context models enable us to discover more regularities in the data by querying the context of sounds. However building decision trees repeatedly in the process of searching for the optimal alignments is very time consuming. We have explored several variations of context-based models in an attempt to make the search converge more quickly, without sacrificing quality.

3.4.1 Zero-depth context model

In this variant of the model, during the simulated annealing phase (i.e., when there is some randomness in the search algorithm), the trees are not expanded to their full depth. Instead, for source-level trees, only the root node is calculated and the target level trees are allowed to query only the *itself* position on the source level. Once the simulated annealing reaches the greedy phase, the trees are

[9] We augment the set of possible feature values at every node with two additional special branches: $\neq$ means that the symbol at the queried position is of the wrong **type** and hence does not have the queried feature; # means the query ran past the beginning of the word.

grown in the same way as they would have been normally, without any restrictions.

This model results in reasonable alignments and relatively low costs and lower running time.

3.4.2 Infinite-depth context model

This is another restrictive variation of the context model, which is more permissive than the zero-depth model. In this variation during the simulated annealing phase of the algorithm, the candidates that can be queried to expand the root nodes of the trees are limited to already encoded features of the *itself* position.

4 Evaluation

We discuss two views on evaluation—*strict* evaluations vs. *intuitive* evaluations.

4.1 Comparing context models to each other

From a strictly information-theoretic point of view, a sufficient condition to claim that model M_1 is better than M_2, is that M_1 assigns a higher probability (equivalently—lower cost) to the observed data. Figure 7A shows the absolute costs, *in bits*, for all language pairs—for the baseline 1-1 model and six context models. The six context models are: the "normal" model, zero-depth and infinite-depth—and for each, the objective function uses either NML or prequential coding.

Here is how we interpret the points in these scatter plots. Each box in the triangular plot compares one model, M_x—whose scores are plotted on the X-axis—against another model, M_y (on the Y-axis). For example, the leftmost column compares the baseline 1-1 model as M_x against each of the six context models in turn; etc. In every plot box, each of the 10×9 points is a comparison of the two models M_x and M_y on one language pair (L_1, L_2). Therefore, for each point (L_1, L_2), the X-coordinate gives the score of model M_x, and the Y-coordinate gives the score of the other model, M_y. If the point (L_1, L_2) is below the diagonal, M_x has higher cost on (L_1, L_2) than M_y. The further away the point is from the "break-even" diagonal line $x = y$, the greater the advantage of one model over the other.

The left column of figure 7A shows that all context models always produce much lower cost compared to the basic context-free 1-1 model defined in (Wettig et al., 2011).

The remaining five columns compare the context models among themselves. Here we see that

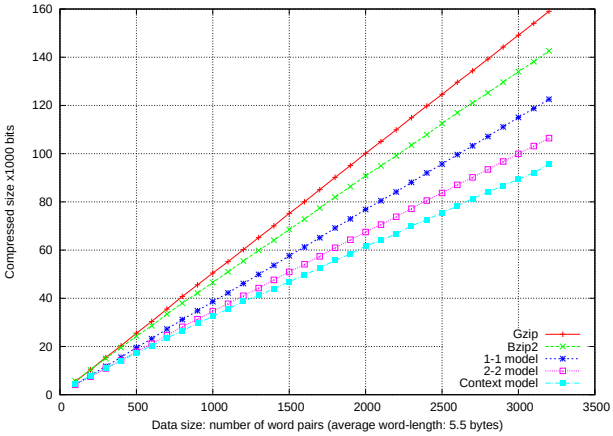

Figure 6: Comparison of compression power

no model variant is a clear winner. Since the variants do not show a clear preference for the "best" context model among this set, we will use *all* of them, to vote as an ensemble.

In figure 6, we compare the context model against standard data compressors, Gzip and Bzip, as well as the baseline models in (Wettig et al., 2011), tested on 3200 Finnish–Estonian data from SSA. Gzip/Bzip compress data by finding regularities—which are frequent sub-strings.

These comparisons confirm that the context model finds more regularity in the data than the off-the-shelf data compressors—which have no knowledge that the words in the data are genetically related—as well as the 1-1 and 2-2 models.

4.2 Imputation

Strictly, the improvement in the compression cost is adequate proof that the presented model outperforms the baselines. For a more intuitive evaluation of improvement in model quality, we can compare models by using them to *impute* unseen data. This is done as follows.

For a given model M, and a language pair (L_1, L_2)—e.g., (Finnish, Estonian)—we hold out one word pair, and train the model on all remaining word pairs. Then we show the model the held out Finnish word and let it impute—i.e., guess—the corresponding Estonian word. Imputation can be done for all models with a dynamic programming algorithm, similar to the Viterbi-like search used during model training. Formally, given the held-out Finnish string, the imputation procedure selects—from *all* possible Estonian strings—the most probable Estonian string, given the model.

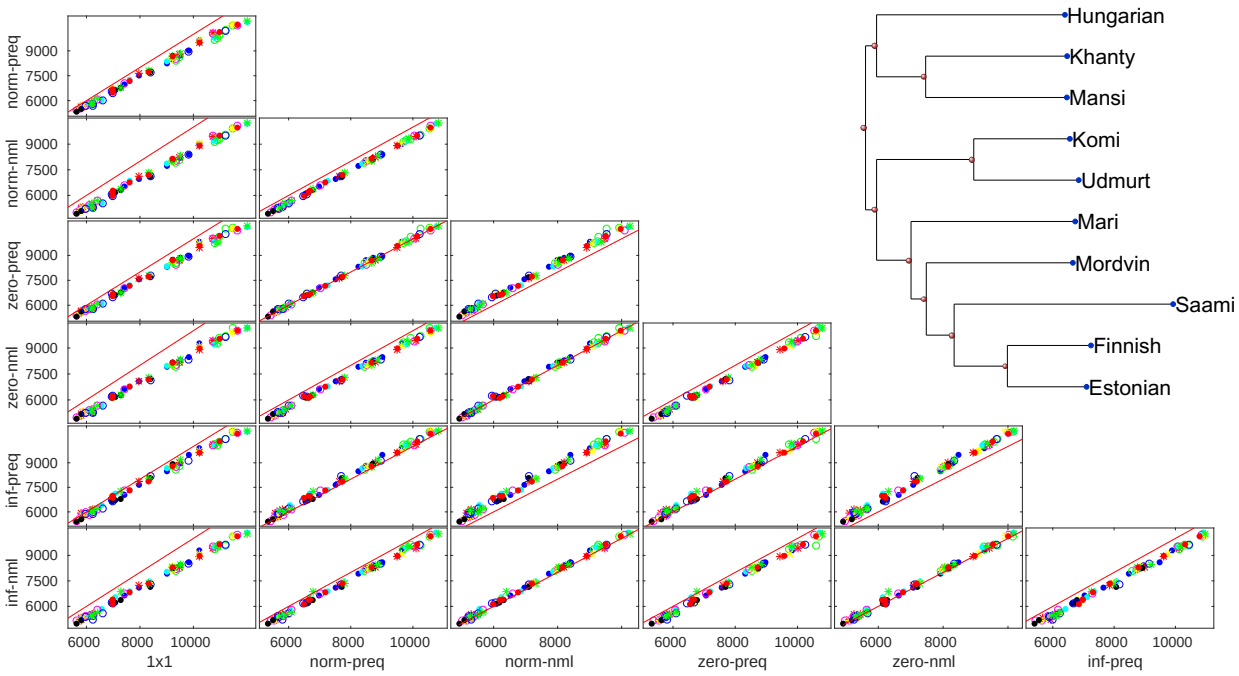

Figure 7: *(A: left) Comparison of costs of context models and the baseline 1-1; (B: upper right) Finno-Ugric tree induced by imputation and normalized edit distances, via NeighborJoin*

We then compute an edit distance (e.g., the Levenshtein edit distance) between the imputed Estonian string and the correct withheld word.

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 correct L_2 data—giving the *Normalized Edit Distance*: $NED(L_2|L_1, M)$ from L_1 to L_2, under M.

NED indicates how much regularity the model has learned about the language pair (L_1, L_2). Finally, we used NED to compare models across all language pairs. The context models always have lower cost than the baseline, and lower NED in $\approx$88% of the language pairs. This is encouraging indication that optimizing the code length is a good approach: the models *do not* optimize NED directly, and yet the cost correlates with NED—a simple and intuitive measure of model quality.

A similar kind of imputation was used in (Bouchard-Côté et al., 2007) for cross-validation.

4.3 Voting for phylogenies

Each context model assigns its own MDL cost to every language pair. These raw MDL costs are not directly comparable, since different language pairs have different amounts of data—different number of shared cognate words. We can make these costs comparable by normalizing them, using *NCD*—Normalized Compression Distance, (Cilibrasi and Vitanyi, 2005), as in (Wettig et al., 2011). Then, each model produces its own pairwise distance matrix for *all* language pairs—where the distance is NCD. A pairwise distance matrix can be used to construct a phylogeny for the language family.

NED, introduced above, provides yet another *distance measure* between any pair of languages, similarly to NCD. Thus, the NED scores can also be used to make inferences about how far the languages are from each other, and used as in put to algorithms for creating phylogenetic trees. For example, applying the *NeighborJoin* algorithm, (Saitou and Nei, 1987), to the pairwise NED matrix produced by the normal context model, yields the phylogeny in Figure 7B.

To compute how far a given phylogeny is from a gold-standard tree, we can use a distance measure for unrooted, leaf-labeled (URLL) trees. One such URLL distance measure is given in (Robinson and Foulds, 1981). The URLL distance between this tree and the gold standard in Figure 1 is 0.12.[10]

However, the MDL costs do not allow us to prefer any one of the context models over the others.

[10]This URLL distance of 0.12 is also quite small. We computed the *expected* URLL distance from a *random* tree with this leaf set over a sample of 1000 randomly generated trees—which is over 0.8. The number of leaf-labeled trees with n nodes is $(2n-3)!!$ (see, e.g., (Ford, 2010)).

Model	Brit.	Ant.	Volga
normal-nml-avg.NCD	0.14	0	0.14
normal-nml-avg.NED	0.14	0	0.14
normal-nml-min.NCD	0.14	0	0.14
normal-nml-min.NED	0.28	0.14	0.28
normal-prequential-avg.NCD	0.14	0	0.14
normal-prequential-avg.NED	0.14	0.28	0.42
normal-prequential-min.NCD	0.14	0	0.14
normal-prequential-min.NED	0.14	0.28	0.42
∞-nml-avg.NCD	0.28	0.14	0.28
∞-nml-avg.NED	0.42	0.28	0.42
∞-nml-min.NCD	0.28	0.14	0.28
∞-nml-min.NED	0.28	0.14	0.28
∞-prequential-avg.NCD	0.14	0	0.14
∞-prequential-avg.NED	0.28	0.14	0.28
∞-prequential-min.NCD	0.14	0.28	0.42
∞-prequential-min.NED	0.28	0.14	0.28
zero-nml-avg.NCD	0.42	0.42	0.57
zero-nml-avg.NED	0	0.14	0.28
zero-nml-min.NCD	0.14	0	0.14
zero-nml-min.NED	0.28	0.28	0.42
zero-prequential-avg.NCD	0.14	0	0.14
zero-prequential-avg.NED	0.28	0.14	0.28
zero-prequential-min.NCD	0.14	0	0.14
zero-prequential-min.NED	0.28	0.28	0.42
Total vote	5.14	**3.28**	6.71

Table 1: Context models voting for Britannica, Anttila and Volga gold standards

Therefore, we use all models as an ensemble.

Gold-standard trees: Different linguists advocate different, conflicting theories about the structure of the Uralic family tree, and Finno-Ugric in particular. Figure 1 shows one such phylogeny, we call "Britannica." Another phylogeny, isomorphic to the tree in Figure 7B, we call "Anttila." A third tree in the literature pairs Mari and Mordvin together into a "Volgaic" branch of Finno-Ugric.

In Table 1, we compare trees generated by the context models to these three gold-standard trees, using the URLL distance defined above.

The context models induce phylogenetic trees as follows. Each model can use prequential coding or NML. Each model yields one NCD matrix and one NED matrix. Finally, for any pair of languages L_1 and L_2, the model in general produces different distances for (L_1, L_2) vs. (L_2, L_1), depending on which language is the source and which is the target (since some languages preserve more information than others). Therefore, each of the three context models produces 8 trees, 24 in total. The distance from each tree to the three gold-standard phylogenies is in Table 1.

The measures show which gold-standard tree is favored by all models taken together. The models strongly prefer "Anttila"—which happens to be the phylogeny favored by a majority of Uralic scholars at present, (Anttila, 1989).

5 Discussion and future work

We have presented an approach to modeling evolutionary processes within a language family by coding data from all languages pair-wise. To our knowledge, these models represent the first attempt to capture *longer-range* context in evolutionary modeling, where prior work allowed small neighboring context to condition the correspondences. We present a feature-based context-aware MDL coding scheme, and compare it against our earlier models, in terms of compression cost and imputation power. Language distances induced by compression cost and by imputation for all pairs of languages, enable us to build complete phylogenies. The model takes a set of lexical data as input, and makes no further assumptions. In this regard, it is as objective as possible given the data.[11]

Finally, we note that our experiments with the context models confirm that the notion of alignment is secondary in modeling evolution. In the old approach, we aligned symbols *jointly*, and hoped to find symbol pairs that align to each other frequently. In the new approach, we code symbols separately one by one on the source and target level, and A. we code the symbols one feature at a time, and B. while coding each feature, we allow the model to use information from any feature of any symbol that has been coded previously.

These models do better, with no alignment.

The objectivity of models given the data opens new possibilities for comparing entire data sets. For example, we can begin to compare the Finnish/Estonian data in StarLing vs. other datasets—and the comparison will be impartial, relying solely on the given data. The models also enable us to quantify the uncertainty of individual entries in the corpus of etymological data. For example, for a given entry x in language L_1, we can compute 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 in particular to entries marked as dubious by the database creators.

[11] The data set itself, of course, may be highly subjective. Refining the data set is in itself an important challenge, as presented in problem a. in the Introduction, to be addressed in future work.

Acknowledgments

This research was supported in part by the Uralink Project and the FinUgRevita Project of the Academy of Finland, and by the National Centre of Excellence "ALGODAN: Algorithmic Data Analysis" of the Academy of Finland. We thank Teemu Roos for his assistance. We are grateful to the anonymous reviewers for their comments and suggestions.

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.

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

Alexandre Bouchard-Côté, Thomas L. Griffiths, and Dan Klein. 2009. Improved reconstruction of protolanguage word forms. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL09)*.

Rudi Cilibrasi and Paul M.B. Vitanyi. 2005. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545.

Daniel J. Ford. 2010. Encodings of cladograms and labeled trees. *Electronic Journal of Combinatorics*, 17:1556–1558.

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

David Hall and Dan Klein. 2011. Large-scale cognate recovery. In *Empirical Methods in Natural Language Processing (EMNLP)*.

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: 19^{th} International Conference on Computational Linguistics*, pages 488–494, Taipei.

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. 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. 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.

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.

Javad Nouri, Jukka Sirén, Jukka Corander, and Roman Yangarber. 2016. From alignment of etymological data to phylogenetic inference via population genetics. In *Proceedings of CogACLL: the 7^{th} Workshop on Cognitive aspects of Computational Language Learning, at ACL-2016*, Berlin, Germany, August. Association for Computational Linguistics.

Károly Rédei. 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.

D.F. Robinson and L.R. Foulds. 1981. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1–2):131–147.

Naruya Saitou and Masatoshi Nei. 1987. The Neighbor-Joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425.

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

Sergei A. Starostin. 2005. Tower of Babel: StarLing etymological databases. http://newstar.rinet.ru/.

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

Hannes Wettig, Kirill Reshetnikov, and Roman Yangarber. 2012. Using context and phonetic features in models of etymological sound change. In *Proc. EACL Workshop on Visualization of Linguistic Patterns and Uncovering Language History from Multilingual Resources*, pages 37–44, Avignon, France.