Measuring Language Closeness by Modeling Regularity

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

Abstract

This paper addresses the problems of measuring similarity between languages—where the term *language* covers any of the senses denoted by *language*, *dialect* or *linguistic variety*, as defined by any theory. We argue that to devise an effective way to measure the similarity between languages one should build a probabilistic model that tries to capture as much regular correspondence between the languages as possible. This approach yields two benefits. First, given a set of language data, for any two models, this gives a way of objectively determining which model is better, i.e., which model is more likely to be accurate and informative. Second, given a model, for any two languages we can determine, in a principled way, how close they are. The better models will be better at judging similarity. We present experiments on data from three language families to support these ideas. In particular, our results demonstrate the arbitrary nature of terms such as *language* vs. *dialect*, when applied to related languages.

1 Introduction

In the context of building and applying NLP tools to similar languages, language varieties, or dialects,[1] we are interested in principled ways of capturing the notion of language *closeness*.

Starting from scratch to develop resources and tools for languages that are close to each other is expensive; the hope is that the cost can be reduced by making use of pre-existing resources and tools for related languages, which are richer in resources.

In the context of this workshop, we assume that we deal with some method, "Method X," that is applied to two (or more) related languages. For example, Method X may involve adapting/porting a linguistic resource from one language to another; or may be trying to translate between the languages; etc. We also assume that the success of Method X directly depends in some way on how similar—or close—the languages are: that is, the similarity between the languages is expected to be a good predictor of how successful the application of the method will be. Thus, in such a setting, it is worthwhile to devote some effort to devising good ways of measuring similarity between languages. This is the main position of this paper.

We survey some of the approaches to measuring inter-language similarity in Section 2. We assume that we are dealing with languages that are *related* genetically (i.e., etymologically). Related languages may be (dis)similar on many levels; in this paper, we focus on similarity on the lexical level. This is admittedly a potential limitation, since, e.g., for Method X, similarity on the level of syntactic structure may be more relevant than similarity on the lexical level. However, as is done in other work, we use lexical similarity as a "general" indicator of relatedness between the languages.[2]

Most of the surveyed methods begin with *alignment* at the level of individual phonetic segments (phones), which is seen as an essential phase in the process of evaluating similarity. Alignment procedures are applied to the input data, which are sets of words which are judged to be similar (cognate)—drawn from the related languages.

Once an alignment is obtained using some method, the natural question arises: how effective is the particular output alignment?

Once the data is aligned (and, hopefully, aligned

[1] We use the term *language* to mean any of: *language*, *dialect*, or *linguistic variety*, according to any definition.

[2] This is a well-studied subject in linguistics, with general consensus that the lexical level has stronger resistance to change than other levels.

Language Technology for Closely Related Languages and Language Variants (LT4CloseLang), pages 56–65,
October 29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

well), it becomes possible to devise measures for computing *distances* between the aligned words. One of the simplest of such measures is the Levenshtein edit distance (LED), which is a crude count of edit operations needed to transform one word into one another. Averaging across LEDs between individual word pairs gives an estimate of the distance between the languages. The question then arises: how accurate is the obtained distance?

LED has obvious limitations. LED charges an edit operation for substituting similar as well as dissimilar phones—regardless of how regular (and hence, probable) a given substitution is. Conversely, LED charges nothing for substituting a phone *x* in language A for the same phone in language B, even if *x* in A *regularly* (e.g., always!) corresponds to *y* in B. More sophisticated variants of LED are then proposed, which try to take into account some aspects of the natural alignment setting (such as assigning different weights to different edit operations, e.g., by saying that it is cheaper to transform *t* into *d* than *t* into *w*).

Thus, in pursuit of effective similarity measures, we are faced with a sequence of steps: procedures for aligning data produce alignments; from the individual word-level alignments we derive distance measures; averaging distances across all words we obtain similarity measures between languages; we then require methods for comparing and *validating* the resulting language distance measures. At various phases, these steps involve *subjectivity*—typically in the form of gold standards. We discuss the kinds of subjectivity encountered with this approach in detail in Section 2.1.

As an alternative approach, we advocate viewing closeness between languages in terms of *regularity* in the data: if two languages are very close, it means that either the differences between them are very few, or—if they are many—then they are very regular.[3] As the number of differences grows and their nature becomes less regular, the languages grow more distant. The goal then is to build probabilistic models that capture regularity in the data; to do this, we need to devise algorithms to discover as much regularity as possible.

This approach yields several advantages. First, a model assigns a probability to observed data. This has deep implications for this task, since it allows us to quantify uncertainty in a principled fashion, rather than commit to ad-hoc decisions and prior assumptions. We will show that probabilistic modeling requires us to make fewer subjective judgements. Second, the probabilities that the models assign to data allow us to build natural distance measures. A pair of languages whose data have a higher probability under a given model are closer than a pair with a lower probability, in a well-defined sense. This also allows us to define distance between individual word pairs. The smarter the model—i.e., the more regularity it captures in the data—the more we will be able to trust in the distance measures based on the model. Third—and equally important for this problem setting—this offers a principled way of comparing methods: if model X assigns higher probability to real data than model Y, then model X is better, and can be trusted more. The key point here is that we can then compare models without any "ground truth" or gold-standard, pre-annotated data.

One way to see this is by using the model to predict *unobserved* data. We can withhold one word pair (w_A, w_B) from languages A and B before building the model (so the model does not see the true correspondence); once the model is built, show it w_A, and ask what is the corresponding word in B. Theoretically, this is simple: the best guess for $\hat{w}_B$ is simply the one that maximizes the probability of the pair $p_M(w_A, \hat{w}_B)$ under the model, over *all* possible strings $\hat{w}_B$ in B.[4] Measuring the distance between w_B and $\hat{w}_B$ tells how good M is at predicting unseen data. Now, if model M_1 consistently predicts better than M_2, it is very difficult to argue that M_1 is in any sense the worse model; and it is able to predict better only because it has succeeded in learning more about the data and the regularities in it.

Thus we can compare different models for measuring linguistic similarity. And this can be done in a principled fashion—if the distances are based on probabilistic models.

The paper is organized as follows. We continue with a discussion of related work. In Section 3 we present one particular approach to modeling, based on information-theoretic principles. In Section 4 we show some applications of these models to several linguistic data sets, from three different language families. We conclude with plans for fu-

[3] In the former case, the differences form a short list; in the latter, the *rules* describing the differences form a short list.

[4] In practice, this can be done efficiently, using heuristics to constrain the search over all strings $\hat{w}_B$ in B.

ture work, in Section 5.

2 Related work

In this section we survey related work on similarity measures between languages, and contrast the principles on which this work relies against the principles which we advocate.

2.1 Subjectivity

Typically, alignment-based approaches use several kinds of inputs that have a subjective nature.

One such input is the data itself, which is to be aligned. For a pair of closely related dialects, deciding which words to align may appear "self-evident." However, as we take dialects/languages that are progressively more distant, such judgements become progressively less self-evident; therefore, in all cases, we should keep in mind that the input data itself is a source of subjectivity in measuring similarity based on data that is comprised of lists of related words.

Another source of subjectivity in some of the related work is *gold-standard* alignments, which accompany the input data. Again, for very close languages, the "correct" alignment may appear to be obvious. However, we must recognize that this necessarily involves subjective judgements from the creators of the gold-standard alignment.

Further, many alignment methods pre-suppose one-to-one correspondence between phones. On one hand, this is due to limitations of the methods themselves (there exist methods for aligning phones in other than one-to-one fashion); on another hand, it violates accepted linguistic understanding that phones do not need to correspond in a one-to-one fashion among close languages. Another potential source of subjectivity comes in the form of prior assumptions or restrictions on permissible alignments.[5] Another common assumption is insistence on consonant-to-consonant and vowel-to-vowel alignments. More relaxed assumptions may come in the form of prior probabilities of phone alignments. Although these may appear "natural" in some sense, it is important to keep in mind that they are *ad hoc*, and reflect a subjective judgement which may not be correct.

After alignment and computation of language distance, the question arises: which of the distance measures is more accurate? Again, one way to answer this question is to resort to gold standards. For example, this can be done via phylogenetic clustering; if method A says language l_1 is closer to l_2 than to l_3, and method B says the opposite (that l_1 is closer to l_3), and if we "know" the latter to be true—from a gold standard—then we can prefer method B. Further, if we have a gold-standard tree for the group of languages, we can apply tree-distance measures[6] to check how the trees generated by a given method differ from the gold-standard. The method that deviates least from the gold standard is then considered best.

2.2 Levenshtein-based algorithms

The Levenshtein algorithm is a dynamic programming approach for aligning a word pair (A, B) using a least expensive set of insertion, deletion and substitution operations required for transforming A into B. While the original Levenshtein edit distance is based on these three operations without any restrictions, later algorithms adapt this method by additional edit operations or restrictions.

Wieling et al. (2009) compare several alignment algorithms applied to dialect pronunciation data. These algorithms include several adaptations of the Levenshtein algorithm and the Pair Hidden Markov Model. They evaluate the algorithms by comparing the resulting pairwise alignments to alignments generated from a set of manually corrected multiple alignments. Standard Levenshtein edit distance is used for comparing the output of each algorithm to the gold standard alignment, to determine which algorithm is preferred.

All alignment algorithms based on Levenshtein distance evaluated by Wieling et al. (2009) restrict aligning vowels with consonants.

VC-sensitive Levenshtein algorithm: uses the standard Levenshtein algorithm, prohibits aligning vowels with consonants, and assigns unit cost for all edit operations. The only sense in which it captures *regularities* is the assumption that the same symbol in two languages represents same sound, which results in assigning a cost of 0 to aligning a symbol to itself. It also prevents the algorithm from finding vowel-to-consonant correspondences (found in some languages), such as *u–v*, *u–l*, etc.

Levenshtein algorithm with Swap: adds an edit operation to enable the algorithm to capture phenomena such as metathesis, via a *transposition*:

[5] One-to-one alignment is actually one such restriction.

[6] Tree-distance measures are developed in the context of work on phylogenetic trees in biological/genetic applications.

aligning ab in A to ba in B costs a single edit operation. This algorithm also forbids aligning vowels to consonants, except in a swap.

Levenshtein algorithm with generated segment distances based on phonetic features: The above algorithms assign unit cost for all edit operations, regardless of how the segments are related. Heeringa (2004) uses a variant where the distances are obtained from differences between phonetic features of the segment pairs. The authors observe that this is subjective because one could choose from different possible feature sets.

Levenshtein algorithm with generated segment distances based on acoustic features: To avoid subjectivity of feature selection, Heeringa (2004) experiments with assigning different costs to different segment pairs based on how phonetically close they are; segment distances are calculated by comparing *spectrograms* of recorded pronunciations. These algorithms do not attempt to discover regularity in data, since they only consider the word pair at a time, using no information about the rest of the data.

Levenshtein algorithm with distances based on PMI: Wieling et al. (2009) use Point-wise Mutual Information (PMI) as the basis for segment distances. They assign different costs to segments, and use the entire dataset for each alignment. PMI for outcomes x and y of random variables X and Y is defined as:

$$pmi(x, y) = \log_2 \frac{p(x, y)}{p(x)p(y)} \quad (1)$$

PMI is calculated using estimated probabilities of the events. Since greater PMI shows higher tendency of x and y to co-occur, it is reversed and normalized to obtain a dissimilarity measure to be used as segment distance. Details about this method are in (Wieling and Nerbonne, 2011).

2.3 Other distance measures

Ellison and Kirby (2006) present a distance measure based on comparing intra-language lexica only, arguing that there is no well-founded common language-independent phonetic space to be used for comparing word forms across languages. Instead, they focus on inferring the distances by comparing how meanings in language A are likely to be confused for each other, and comparing it to the confusion probabilities in language B.

Given a lexicon containing mappings from a set of meanings M to a set of forms F, confusion probability $P(m_1|m_2; L)$ for each pair of meanings (m_1, m_2) in L is the probability of confusing m_1 for m_2. This probability is formulated based on an adaptation of *neighborhood activation model*, and depends on the edit distance between the corresponding forms in the lexicon. Following this approach, they construct a confusion probability matrix for each language, which can be viewed as a probability distribution. Inter-language distances are then calculated as the distance between the corresponding distributions, using symmetric Kullback-Liebler distance and Rao distance . The inferred distances are used to construct a phylogenetic tree of the Indo-European languages. The approach is evaluated by comparing the resulting taxonomy to a gold-standard tree, which is reported to be a good fit.

As with other presented methods, although this method can be seen as measuring distances between languages, there remain two problems. First, they do not reflect the genetic differences and similarities—and regularities—between the languages in a transparent, easily interpretable way. Second, they offer no direct way to compare competing approaches, except indirectly, and using (subjective) gold-standards.

3 Methods for measuring language closeness

We now discuss an approach which follows the proposal outlined in Section 1, and allows us to build probabilistic models for measuring closeness between languages. Other approaches that rely on probabilistic modeling would serve equally well. A comprehensive survey of methods for measuring language closeness may be found in (Wieling and Nerbonne, 2015). Work that is probabilistically oriented, similarly to our proposed approaches, includes (Bouchard-Côté et al., 2007; Kondrak, 2004) and others. We next review two types of models (some of which are described elsewhere), which are based on information-theoretic principles. We discuss how these models suit the proposed approach, in the next section.

3.1 1-1 symbol model

We begin with our "basic" model, described in (Wettig and Yangarber, 2011; Wettig et al., 2011), which makes several simplifying assumptions—which the subsequent, more advanced models relax (Wettig et al., 2012; Wettig

et al., 2013).[7] The basic model is based on alignment, similarly to much of the related work mentioned above: for every word pair in our data set—the "corpus"—it builds a complete alignment for all symbols (Wettig et al., 2011). The basic model considers *pairwise* alignments only, i.e., two languages at a time; we call them the *source* and the *target* languages. Later models relax this restriction by using N-dimensional alignment, with $N > 2$ languages aligned simultaneously. The basic model allows only *1-1* symbol alignments: one source symbol[8] may correspond to one target symbol—or to the empty symbol ϵ (which we mark as "."). More advanced models align substrings of more than one symbol to each other. The basic model also ignores *context*, whereas in reality symbol correspondences are heavily conditioned on their context. Finally, the basic model treats the symbols as *atoms*, whereas more advanced models treat the symbols as vectors of distinctive features.

We distinguish between the raw, *observed* data and *complete* data—i.e., complete with the alignment; the *hidden* data is where the insertions and deletions occur. For example, if we ask what is the "correct" alignment between Finnish *vuosi* and Khanty *al* (cognate words from these two Uralic languages, both meaning "year"):

v	u	o	.	s	i		v	u	o	s	i
|	|	|	|	|	|		|	|	|	|	|
.	a	.	l	.	.		.	.	a	l	.

are two possible alignments, among many others. From among all alignments, we seek the *best* alignment: one that is *globally* optimal, i.e., one that is consistent with as many regular sound correspondences as possible. This leads to a chicken-and-egg problem: on one hand, if we had the best alignment for the data, we could simply read off a set of rules, by observing which source symbol corresponds frequently to which target symbol. On the other hand, if we had a complete set of rules, we could construct the best alignment, by using dynamic programming (*à la* one of the above mentioned methods, since the costs of all possible edit operations are determined by the rules). Since at the start we have neither, the rules and the alignment are bootstrapped in tandem.

Following the Minimum Description Length (MDL) principle, the best alignment is the one that can be *encoded* (i.e., written down) in the shortest space. That is, we aim to code the complete data—for all word pairs in the given language pair—as compactly as possible. To find the optimal alignment, we need A. an objective function—a way to measure the quality of any given alignment—and B. a search algorithm, to sift through all possible alignments for one that optimizes the objective.

We can use various methods to code the complete data. Essentially, they all amount to measuring how many bits it costs to "transmit" the complete set of alignment "events", where each alignment event e is a pair of aligned symbols $(\sigma : \tau)$

$$e = (\sigma : \tau) \in \Sigma \cup \{., \#\} \times T \cup \{., \#\}$$

drawn from the source alphabet Σ and the target alphabet T, respectively.[9] One possible coding scheme is "prequential" coding, or the Bayesian marginal likelihood, see, e.g., (Kontkanen et al., 1996), used in (Wettig et al., 2011); another is normalized maximum likelihood (NML) code, (Rissanen, 1996), used in (Wettig et al., 2012).

Prequential coding gives the total code length

$$L_{base}(D) = -\sum_{e \in E} \log c(e)! + \log \left[\sum_{e \in E} c(e) + K - 1 \right]! - \log(K-1)! \quad (2)$$

for data D. Here, $c(e)$ denotes the event count, and K is the total number of event types.

To find the optimal alignments, the algorithm starts with aligning word pairs randomly, and then iteratively searching for the best alignment given rest of the data for each word pair at a time. To do this, we first exclude the current alignment from our *complete* data. The best alignment in the realigning process is found using a *Dynamic Programming* matrix, with source word symbols in the rows and target word symbols as the columns. Each possible alignment of the word pair corresponds to a path from top-left cell of the matrix to the bottom-right cell. Each cell $V(\sigma_i, \tau_j)$ holds the cost of aligning sub-string $\sigma_1..\sigma_i$ with $\tau_1..\tau_j$, and is computed as:

$$V(\sigma_i, \tau_j) = \min \begin{cases} V(\sigma_i, \tau_{j-1}) & +L(. : \tau_j) \\ V(\sigma_{i-1}, \tau_j) & +L(\sigma_i : .) \\ V(\sigma_{i-1}, \tau_{j-1}) & +L(\sigma_i : \tau_j) \end{cases} \quad (3)$$

[7] The models can be downloaded from *etymon.cs.helsinki.fi*

[8] In this paper, we equate *symbols* with *sounds*: we assume our data to be given in *phonetic* transcription.

[9] Note, that the alphabets need not be the same, or even have any symbols in common. We add a special end-of-word symbol, always aligned to itself: $(\# : \#)$. Empty alignments $(. : .)$ are not allowed.

where $L(e)$ is the cost of coding event e. The cost of aligning the full word pair, is then found in the bottom-right cell, and the corresponding path is chosen as the new alignment, which is registered back into the complete data.

We should mention that due to vulnerability of the algorithm to local optima, we use simulated annealing with (50) random restarts.

3.2 Context model

Context model is described in detail in (Wettig et al., 2013). We use a modified version of this model to achieve faster run-time.

One limitation of the basic model described above is that it uses no information about the context of the sounds, thus ignoring the fact that linguistic sound change is regular and highly depends on context. The 1-1 model also treats symbols of the words as $atoms$, ignoring how two sounds are phonetically close. The context model, addresses both of these issues.

Each sound is represented as a vector of distinctive phonetic features. Since we are using MDL as the basis of the model here, we need to code (i.e., transmit) the data. This can be done by coding one feature at a time on each level.

To code a feature F on a level L, we construct a decision tree. First, we collect all instances of the sounds in the data of the corresponding level that have the current feature, and then build a count matrix based on how many instances take each value. Here is an example of such a matrix for feature *V* (vertical articulation of a vowel).

V	Close	Mid-close	Mid-open	Open
	10	25	33	24

This shows that there are 10 *close* vowels, 25 *mid-close* vowels, etc.

This serves as the root node of the tree. The tree can then query features of the sounds in the current context by choosing from a set of candidate contexts. Each candidate is a triplet (L, P, F), representing *Level*, *Position*, and *Feature* respectively.

L can be either source or target, since we are dealing with a pair of language varieties at a time. P is the position of the sound that is being queried relative to current sound, and F is the feature being queried. Examples of a *Position* are *previous vowel*, *previous position*, *itself*, etc. The tree expands depending on the possible responses to the query, resulting in child nodes with their own count matrix. The idea here is to make the matrices in the child nodes as sparse as possible in order to code them with fewer bits.

This process continues until the tree cannot be expanded any more. Finally the data in each leaf node is coded using *prequential* coding as before with the same cost explained in Equation 2.

Code length for the complete data consists of cost of encoding the trees and the cost of encoding the data given the trees. The search algorithm remains the same as the 1-1 algorithm, but uses the constructed trees to calculate the cost of events.

This method spends much time rebuilding the trees on each iteration; its run-time is very high. In the modified version used in this paper, the trees are not allowed to expand initially, when the model has just started and everything is random due to simulated annealing. Once the simulated annealing phase is complete, the trees are expanded fully normally. Our experiments show that this results in trees that are equally good as the original ones.

3.3 Normalized Compression Distance

The cost of coding the data for a language pair under a model reflects the amount of regularity the model discovered, and thus is a means of measuring the distance between these languages. However the cost also depends on the size of the data for the language pair; thus, a way of normalizing the cost is needed to make them comparable across language pairs. We use "*Normalized Compression Distance*" (NCD), described in (Cilibrasi and Vitanyi, 2005) to achieve this.

Given a model that can compress a language pair (a, b) with cost $C(a, b)$, NCD of (a, b) is:

$$NCD(a,b) = \frac{C(a,b) - \min\big(C(a), C(b)\big)}{\max\big(C(a), C(b)\big)} \qquad (4)$$

Since NCD of different pairs are comparable under the same model, it can be used as a distance measure between language varieties.

3.4 Prediction of unobserved data

The models mentioned above are also able to predict unobserved data as described in Section 1 (Wettig et al., 2013).

For the basic 1-1 model, since no information about the context is used, prediction simply means looking for the most probable symbol in target language for each symbol of w_A. For the context model, a more sophisticated dynamic-programming heuristic is needed to predict the unseen word, (Hiltunen, 2012). The predicted word

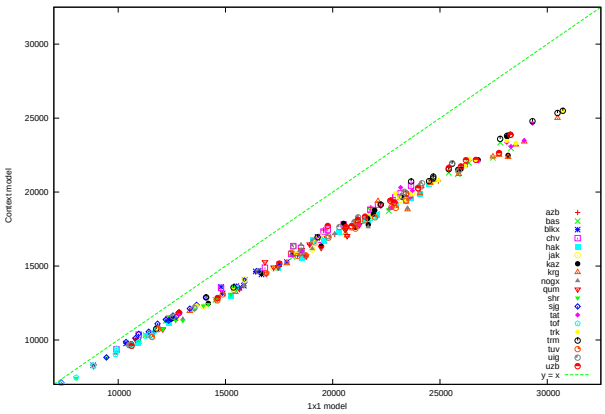

Figure 1: Model comparison: MDL costs.

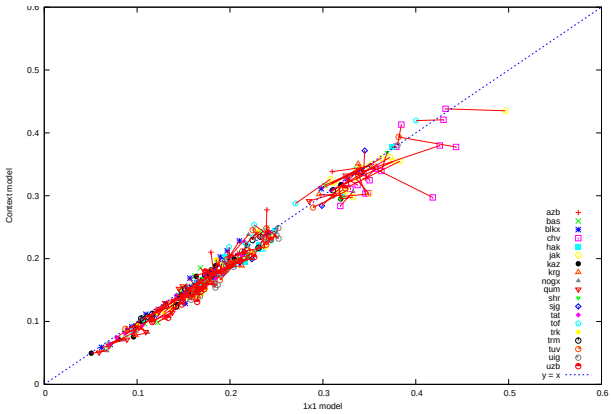

Figure 2: Model comparison: NFED.

$\hat{w}_B$ is then compared to the real corresponding word w_B to measure how well the model performed on the task.

Feature-wise Levenshtein edit distance is used for this comparison. The edit distances for all word pairs are normalized, resulting in *Normalized Feature-wise Edit Distance* (*NFED*) which can serve as a measure of model quality.

4 Experiments

To illustrate the principles discussed above, we experiment with the two principal model types described above—the baseline 1-1 model and the context-sensitive model, using data from three different language families.

4.1 Data

We use data from the StarLing data bases, (Starostin, 2005), for the Turkic and Uralic language families, and for the Slavic branch of the Indo-European family. For dozens of language families, StarLing has rich data sets (going beyond Swadesh-style lists, as in some other lexical data collections built for judging language and dialect distances). The databases are under constant development, and have different quality. Some datasets, (most notably the IE data) are drawn from multiple sources, which use different notation, transcription, etc., and are not yet unified. The data we chose for use is particularly clean.

For the Turkic family, StarLing at present contains 2017 cognate sets; we use 19 (of the total 27) languages, which have a substantial amount of attested word-forms in the data collection.

4.2 Model comparison

We first demonstrate how the "best" model can be chosen from among several models, in a principled way. This is feasible if we work with probabilistic models—models that assign probabilities to the observed data. If the model is also able to perform prediction (of unseen data), then we can measure the model's predictive power and select the best model using predictive power as the criterion. We will show that in the case of the two probabilistic models presented above, these two criteria yield the same result.

We ran the baseline 1-1 model and the context model against the entire Turkic dataset, i.e., the 19×18 language pairs,[10] (with 50 restarts for each pair, a total of 17100 runs). For each language pair, we select the best out of 50 runs for each model, according to the cost it assigns to this language pair. Figure 1 shows the costs obtained by the best run: each point denotes a language pair; X-coordinate is the cost according to the 1-1 model, Y-coordinate is the cost of the context model. The Figure shows that all 19×18 points lie below the diagonal (x=y), i.e., for every language pair, the context model finds a code with lower cost—as is expected, since the context model is "smarter," uses more information from the data, and hence finds more regularity in it.

Next, for each language pair, we take the run that found the lowest cost, and use it to impute unseen data, as explained in Section 3—yielding NFED, the distance from the imputed string to the

[10] Turkic languages in tables and figures are: azb:Azerbaijani, bas:Bashkir, blk:Balkar, chv:Chuvash, hak:Khakas, jak:Yakut, kaz:Kazakh, krg:Kyrgyz, nog:Nogaj, qum:Qumyk, shr:Shor, sjg:Sary Uyghur, tat:Tatar, tof:Tofalar, trk:Turkish, trm:Turkmen, tuv:Tuva, uig:Uyghur, uzb:Uzbek.

	ru	ukr	cz	slk	pl	usrb	lsrb	bulg	scr
ru	0	.41	.41	.39	.41	.51	.53	.48	.40
ukr	.41	0	.48	.46	.51	.49	.50	.48	.47
cz	.40	.48	0	**.29**	.38	.45	.52	.50	.39
slk	.38	.45	**.29**	0	.38	.41	.44	.45	.38
pl	.43	.51	.39	.41	0	.48	.50	.52	.45
usrb	.50	.48	.44	.40	.46	0	**.29**	.49	.48
lsrb	.52	.51	.49	.44	.47	**.30**	0	.52	.50
bulg	.46	.47	.48	.45	.51	.47	.49	0	.41
scr	.40	.47	.38	.38	.43	.49	.51	.44	0

Table 1: NCDs for 9 Slavic languages, StarLing database: context model

actual, correct string in the target language. This again yields 19×18 points, shown in Figure 2; this time the X and Y values lie between 0 and 1, since NFED is normalized. (In the figure, the points are linked with line segments as follows: for any pair (a,b) the point (a,b) is joined by a line to the point (b,a). This is done for easier identification, since the point (a,b) displays the legend symbol for only language a.) Overall, many more points lie below the diagonal, (approximately 10% of the points are above). The context model performs better, and it would therefore be a safer/wiser choice, if we wish to measure language closeness; which agrees with the result obtained using raw compression costs.

The key point here is that this comparison method can accommodate *any* probabilistic model: for any new candidate model we check—over the same datasets—what probability values does the model assign to each data point. Probabilities and (compression) costs are interchangeable: information theory tells us that for a data set D and model M, the probability P of data D under model M and the cost (code length) L of D under M are related by: $L_M(D) = -\log P_M(D)$. If the new model assigns higher probability (or lower cost) to observed data, it is preferable—*obviating the need for gold-standards*, or subjective judgements.

4.3 Language closeness

We next explore various datasets using the context model—the better model we have available.

Uralic: We begin with Uralic data from StarLing.[11] The Uralic database contains data from more than one variant of many languages: we extracted data for the top two dialects—in terms of counts of available word-forms—for Komi, Udmurt, Mari, Mordva, Mansi, Khanty and Saami. Table 2 shows the normalized compression distances for each of the pairs; the NCD costs for Finnish and Estonian are given for comparison.

Language pair		NCD
kom_s	kom_p	.18
kom_p	kom_s	.19
udm_s	udm_g	.20
udm_g	udm_s	.21
mar_b	mar_kb	.28
mar_kb	mar_b	.28
mrd_m	mrd_e	.29
mrd_e	mrd_m	.29
est	fin	.32
fin	est	.32
man_p	man_so	.34
khn_v	khn_dn	.35
khn_dn	khn_v	.36
man_so	man_p	.36
saa_n	saa_l	.37
saa_l	saa_n	.37

Table 2: Comparison of Uralic dialect/language pairs, sorted by NCD: context model.

It is striking that the pairs that score below Finnish/Estonian are all "true" dialects, whereas those that score above are not. E.g., the Mansi variants Pelym and Sosva, (Honti, 1998), and Demjanka and Vakh Khanty, (Abondolo, 1998), are mutually unintelligible. The same is true for North and Lule Saami.

Turkic: We compute NCDs for the Turkic languages under the context model. Some of the Turkic languages are known to form a much tighter dialect continuum, (Johanson, 1998), which is evident from the NCDs in Table 3. E.g., Tofa is most-closely related to the Tuvan language and forms a dialect continuum with it, (Johanson, 1998). Turkish and Azerbaijani closely resemble each other and are mutually intelligible. In the table we highlight language pairs with NCD ≤ 0.30.

Slavic: We analyzed data from StarLing for 9 Slavic languages.[12] The NCDs are shown in Table 1. Of all pairs, the normalized compression costs for (cz, slk) and (lsrb, usrb) fall below the .30 mark, and indeed these pairs have high mutual intelligibility, unlike all other pairs.

When the data from Table 1 are fed into the NeighborJoining algorithm, (Saitou and Nei, 1987), it draws the phylogeny in Figure 3, which clearly separates the languages into the 3 accepted branches of Slavic: East (ru, ukr), South

[11] We use data from the Finno-Ugric sub-family. The language codes are: est:Estonian, fin:Finnish, khn:Khanty, kom:Komi, man:Mansi, mar:Mari, mrd:Mordva, saa:Saami, udm:Udmurt.

[12] The Slavic languages from StarLing: bulg:Bulgarian, cz:Czech, pl:Polish, ru:Russian, slk:Slovak, scr:Serbo-Croatian, ukr:Ukrainian, lsrb/usrb:Lower and Upper Sorbian.

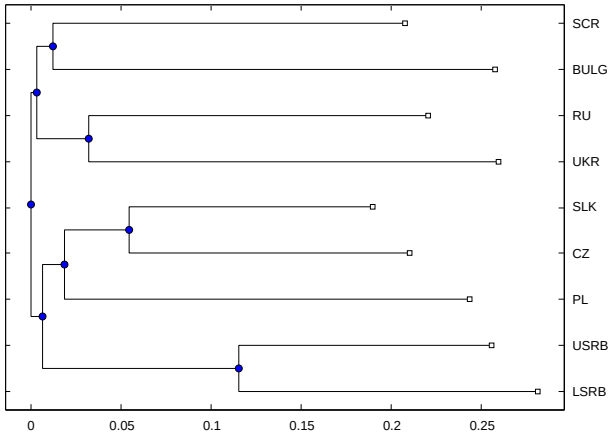

Figure 3: NeighborJoining tree for Slavic languages in Table 1.

(scr, bulg) and West (pl, cz, slk, u/lsrb). The phylogeny also supports later separation (relative time depth > 0.05) of the pairs with higher mutual intelligibility—Upper/Lower Sorbian, and Czech/Slovak.[13]

5 Conclusions and future work

We have presented a case for using probabilistic modeling when we need reliable quantitative measures of language closeness. Such needs arise, for example, when one attempts to develop methods whose success directly depends on how close the languages in question are. We attempt to demonstrate two main points. One is that using probabilistic models provides a principled and natural way of comparing models—to determine which candidate model we can trust more when measuring how close the languages are. It also lets us compare models without having to build gold-standard datasets; this is important, since gold-standards are subjective, not always reliable, and expensive to produce. We are really interested in regularity, and the proof of the model's quality is in its ability to assign high probability to observed and unobserved data.

The second main point of the paper is showing how probabilistic models can be employed to measure language closeness. Our best-performing model seems to provide reasonable judgements of closeness when applied to languages/linguistic variants from very different language families. For all of Uralic, Turkic and Slavic data, those that fell below the 0.30 mark on the NCD axis are known to have higher mutual intelligibility, while those that are above the mark have lower or no mutual intelligibility. Of course, we do not claim that 0.30 is a magic number; for a different model the line of demarcation may fall elsewhere entirely. However, it shows that the model (which we selected on the basis of its superiority according to our selection criteria) is quite *consistent* in predicting the degree of mutual intelligibility, overall.

Incidentally, these experiments demonstrate, in a principled fashion, the well-known arbitrary nature of the terms language vs. dialect—this distinction is simply not supported by real linguistic data. More importantly, probabilistic methods require us to make fewer subjective judgements, with no *ad hoc* priors or gold-standards, which in many cases are difficult to obtain and justify—and rather rely on the observed data as the ultimate and sufficient truth.

[13] We should note that the NCDs produce excellent phylogenies also for the Turkic and Uralic data; not included here due to space constraints.

Acknowledgments

This research was supported in part by: the FinUgRevita Project of the Academy of Finland, and by the National Centre of Excellence "Algorithmic Data Analysis (ALGODAN)" of the Academy of Finland.

References

Daniel Abondolo. 1998. Khanty. In Daniel Abondolo, editor, *The Uralic Languages*, pages 358–386. Routledge.

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.

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

T. Mark Ellison and Simon Kirby. 2006. Measuring language divergence by intra-lexical comparison. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-2006)*, pages 273–280, Sydney, Australia.

Wilbert Heeringa. 2004. *Measuring Dialect Pronunciation Differences using Levenshtein Distance*. Ph.D. thesis, Rijksuniversiteit Groningen.

	azb	bas	blk	chv	hak	jak	kaz	krg	nog	qum	shr	sjg	tat	tof	trk	trm	tuv	uig	uzb
azb	0	.40	.35	.60	.48	.59	.38	.39	.35	.34	.43	.41	.38	.45	**.27**	.33	.46	.40	.37
bas	.39	0	**.28**	.58	.45	.59	**.27**	.31	**.24**	**.26**	.40	.41	**.21**	.46	.39	.38	.45	.36	.34
blk	.35	**.30**	0	.57	.42	.57	**.26**	**.28**	**.22**	**.19**	.36	.40	**.27**	.42	.34	.36	.42	.35	**.30**
chv	.59	.59	.56	0	.62	.67	.56	.58	.55	.53	.60	.57	.56	.60	.56	.60	.62	.61	.58
hak	.47	.44	.41	.63	0	.58	.41	.40	.37	.40	**.27**	.43	.43	.39	.46	.50	.40	.46	.46
jak	.57	.57	.57	.70	.58	0	.56	.57	.55	.54	.55	.54	.57	.51	.58	.57	.56	.58	.57
kaz	.38	**.28**	**.27**	.57	.42	.57	0	**.24**	**.16**	**.24**	.38	.39	**.29**	.44	.37	.39	.41	.36	.33
krg	.38	.31	**.27**	.60	.40	.57	**.23**	0	**.21**	**.26**	.35	.40	.32	.41	.36	.39	.40	.35	.33
nog	.35	**.25**	**.22**	.57	.39	.55	**.15**	**.22**	0	**.19**	.36	.38	**.26**	.43	.33	.35	.41	.35	.31
qum	.34	**.27**	**.19**	.57	.41	.55	**.23**	**.26**	**.19**	0	.35	.37	**.26**	.41	.33	.35	.41	.33	.31
shr	.43	.40	.36	.63	**.28**	.55	.38	.36	.35	.34	0	.40	.40	.36	.43	.44	.38	.42	.42
sjg	.43	.42	.41	.58	.45	.55	.40	.41	.39	.38	.40	0	.42	.44	.43	.43	.43	.41	.41
tat	.36	**.22**	**.27**	.60	.44	.59	**.28**	.32	**.26**	**.26**	.40	.41	0	.45	.38	.38	.45	.36	.33
tof	.47	.45	.42	.61	.39	.50	.42	.42	.42	.41	.36	.42	.45	0	.48	.46	**.24**	.44	.43
trk	**.28**	.40	.35	.58	.48	.59	.37	.36	.33	.34	.43	.42	.39	.47	0	.34	.46	.40	.38
trm	.32	.40	.36	.62	.51	.59	.39	.40	.36	.35	.44	.43	.39	.46	.34	0	.49	.41	.36
tuv	.46	.46	.41	.63	.40	.56	.41	.40	.41	.41	.38	.42	.45	**.23**	.45	.48	0	.45	.46
uig	.40	.39	.34	.60	.49	.58	.36	.36	.36	.33	.43	.40	.38	.45	.41	.42	.46	0	.33
uzb	.37	.36	.31	.60	.48	.58	.34	.34	.32	.32	.43	.41	.34	.44	.38	.36	.47	.33	0

Table 3: Normalized compression distances for 19 Turkic languages (StarLing database): context model

Suvi Hiltunen. 2012. Minimum description length modeling of etymological data. Master's thesis, University of Helsinki.

László Honti. 1998. Ob' Ugrian. In Daniel Abondolo, editor, *The Uralic Languages*, pages 327–357. Routledge.

Lars Johanson. 1998. The history of Turkic. In Lars Johanson & Éva Ágnes Csató, editor, *The Turkic Languages*, pages 81–125. London, New York: Routledge. Classification of Turkic languages (at Turkiclanguages.com).

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, Petri Myllymäki, and Henry Tirri. 1996. Constructing Bayesian finite mixture models by the EM algorithm. Technical Report NC-TR-97-003, ESPRIT NeuroCOLT: Working Group on Neural and Computational Learning.

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

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.

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

Hannes Wettig and Roman Yangarber. 2011. Probabilistic models for alignment of etymological data. In *Proceedings of NoDaLiDa: the 18th Nordic Conference on Computational Linguistics*, Riga, Latvia.

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.

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.

Hannes Wettig, Javad Nouri, Kirill Reshetnikov, and Roman Yangarber. 2013. Information-theoretic modeling of etymological sound change. In Lars Borin and Anju Saxena, editors, *Approaches to measuring linguistic differences*, volume 265 of *Trends in Linguistics*, pages 507–531. de Gruyter Mouton.

Martijn Wieling and John Nerbonne. 2011. Measuring linguistic variation commensurably. In *Dialectologia Special Issue II: Production, Perception and Attitude*, pages 141–162.

Martijn Wieling and John Nerbonne. 2015. Advances in dialectometry. In *Annual Review of Linguistics*, volume 1. To appear.

Martijn Wieling, Jelena Prokić, and John Nerbonne. 2009. Evaluating the pairwise string alignment of pronunciations. In *Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education*, pages 26–34, Athens, Greece.