An automated framework for fast cognate detection and Bayesian phylogenetic inference in computational historical linguistics

Taraka Rama  
Department of Linguistics  
University of North Texas  
taraka.kasi@gmail.com

Johann-Mattis List  
Dep. of Ling. and Cult. Evolution (DLCE)  
MPI-SHH (Jena)  
list@shh.mpg.de

Abstract

We present a fully automated workflow for phylogenetic reconstruction on large datasets, consisting of two novel methods, one for fast detection of cognates and one for fast Bayesian phylogenetic inference. Our results show that the methods take less than a few minutes to process language families that have so far required large amounts of time and computational power. Moreover, the cognates and the trees inferred from the method are quite close, both to gold standard cognate judgments and to expert language family trees. Given its speed and ease of application, our framework is specifically useful for the exploration of very large datasets in historical linguistics.

1 Introduction

Computational historical linguistics is a relatively young discipline which aims to provide automated solutions for those problems which have been traditionally dealt with in an exclusively manual fashion in historical linguistics. Computational historical linguists thus try to develop automated approaches to detect historically related words (called “cognates”; Jäger et al. 2017; List et al. 2017; Rama et al. 2017; Rama 2018a), to infer language phylogenies (“language trees”; Rama et al. 2018; Greenhill and Gray 2009), to estimate the time depths of language families (Rama, 2018b; Chang et al., 2015; Gray and Atkinson, 2003), to determine the homelands of their speakers (Bouckaert et al., 2012; Wichmann et al., 2010), to determine diachronic word stability (Pagel and Meade, 2006; Rama and Wichmann, 2018), or to estimate evolutionary rates for linguistic features (Greenhill et al., 2010).

Despite the general goal of automating traditional workflows, the majority of studies concerned with phylogenetic reconstruction (including studies on dating and homeland inference) still make use of expert judgments to determine cognate words in linguistic datasets, because detecting cognates is usually regarded as hard to automate. The problem of manual annotation is that the process is very time consuming and may show a lack of objectivity, as inter-annotator agreement is rarely tested when creating new datasets. The last twenty years have seen a surge of work in the development of methods for automatic cognate identification. Current methods reach high accuracy scores compared to human experts (List et al., 2017) and even fully automated workflows in which phylogenies are built from automatically inferred cognates do not differ a lot from phylogenies derived from expert’s cognate judgments (Rama et al., 2018).

Despite the growing amount of research devoted to automated word comparison and fully automated phylogenetic reconstruction workflows, scholars have so far ignored the computational effort required to apply the methods to large amounts of data. While the speed of the current workflows can be ignored for small datasets, it becomes a challenge with increasing amounts of data, and some of the currently available methods for automatic cognate detection can only be applied to datasets with maximally 100 languages. Although methods for phylogenetic inference can handle far more languages, they require enormous computational efforts, even for small language families of less than 20 varieties (Kolipakam et al., 2018), which make it impossible for scholars perform exploratory studies in Bayesian frameworks.
In this paper, we propose an automated framework for fast cognate detection and fast Bayesian phylogenetic inference. Our cognate detection algorithm uses an alignment-free technique based on character skip-grams (Järvelin et al., 2007), which has the advantage of neither requiring hand-crafted nor statistically trained matrices of probable sound correspondences to be supplied.¹ Our fast approach to Bayesian inference uses a simulated annealing variant (Andrieu et al., 2003) of the original MCMC algorithm to compute a maximum-a-posteriori (MAP) tree in a very short amount of time.

Testing both our fast cognate detection and our fast phylogenetic reconstruction approach on publicly available datasets, we find that the results presented in the paper are comparable to the alternative, much more time-consuming algorithms currently in use. Our automatic cognate detection algorithm shows results comparable to those achieved by the SCA approach (List, 2014), which is one of the best currently available algorithms that work without inferring regular sound correspondences prior to computation (List et al., 2017). Our automatically inferred MAP trees come close to the expert phylogenies reported in Glottolog (Hammarström et al., 2017), and are at least as good as the phylogenies inferred with MrBayes (Ronquist et al., 2012), one of the most popular programs for phylogenetic inference. In combination, our new approaches offer a fully automated workflow for phylogenetic reconstruction in computational historical linguistics, which is so fast that it can be easily run on single core machines, yielding results of considerable quality in less than 15 minutes for datasets of more than 50 languages.

In the following, we describe the fast cognate detection program in Section 2. We describe both the regular variant of the phylogenetic inference program and our simulated annealing variant in Section 3. We present the results of our automated cognate detection and phylogenetic inference experiments and discuss the results in Section 4. We conclude the paper and present pointers to future work in Section 5.

2 Fast Cognate Detection

Numerous methods for automatic cognate detection in historical linguistics have been proposed in the past (Jäger et al., 2017; List, 2014; Rama et al., 2017; Turchin et al., 2010; Arnaud et al., 2017). Most of them are based on the same general workflow, by which – in a first stage – all possible pairs of words within the same meaning slot of a wordlist are compared with each other in order to compute a matrix of pairwise distances or similarities. In a second stage, a flat cluster algorithm or a network partitioning algorithm is used to partition all words into cognate sets, taking the information in the matrix of word pairs as basis (List et al., 2018b). Differences between the algorithms can be found in the way in which the pairwise word comparisons are carried out, to which degree some kind of pre-processing of the data is involved, or which algorithm for flat clustering is being used.

Since any automated word comparison that starts from the comparison of word pairs needs to calculate similarities or distances for all $\frac{n(n-1)}{2}$ possible word pairs in a given concept slot, the computation cost for all algorithms which employ this strategy exponentially increases with the number of words being compared. If methods additionally require to pre-process the data, for example to search across all language-pairs for language-specific similarities, such as regularly corresponding sounds (List et al., 2017; Jäger et al., 2017), the computation becomes impractical for datasets of more than 100 languages.

A linear time solution was first proposed by Dolgopol'sky (1964). Its core idea is to represent all sound sequences in a given dataset by their consonant classes. A consonant class is hereby understood as a rough partitioning of speech sounds into groups that are conveniently used by historical linguistics when comparing languages (such as velars, [k, g, x], dentals [t, d, θ], or liquids [r, l, u], etc.). The major idea of this approach is to judge all words as cognate whose initial two consonant classes match. Given that the method requires only that all words be converted to their

¹Although Rama (2015) uses skip-grams, the approach in the paper requires hand-annotated data which we intend to overcome in this paper.
Figure 1: Bipartite graph of English daughter, German Tochter, and their corresponding sound-class-based skip-grams of size 5.

first consonant classes, this approach, which is now usually called consonant-class matching approach (CCM, Turchin et al. 2010), is very fast, since its computation costs are linear with respect to the number of words being compared. The task of assigning a given word to a given cognate set is already fulfilled by assigning a word a given string of consonant classes.

The drawback of the CCM approach is a certain lack of accuracy. While being quite conservative when applied to words showing the same meaning, the method likewise misses many valid matches and thus generally shows a low recall. This is most likely due to the fact that the method does not not contain any alignment component. Words are converted to sound-class strings and only complete matches are allowed, while good partial matches can often be observed in linguistic data, as can be seen from the comparison of English daughter, represented as TVTVR in sound classes compared to German Tochter TKTVR.

In order to develop an algorithm for automatic cognate detection which is both fast and shows a rather high degree of accuracy, we need to (1) learn from the strategy employed by the CCM method in avoiding any pairwise word comparison, while – at the same time – (2) avoiding the problems of the CCM method by allowing for a detailed sequence comparison based on some kind alignment techniques. Since the CCM method only compares the first two consonants per word, it cannot identify words like English daughter and German Tochter as cognate, although the overall similarity is obvious when comparing the whole strings.

A straightforward way to account for our two requirements is using skip-grams of sound-class representations and to represent words and sound-class skip-grams in a given dataset in form of a bipartite network, in which words are assigned to one type of node, and skip-grams to another one. In such a network, we could compute multiple representations of TVTVR and TVKTVR directly and later see, in which of them the two sequences match. If, for example, we computed all n-grams of length 5 allowing to skip one, we would receive TVTVR for English (only possible solution) and VTKTR, TKTV, TVTVR, TVKVR, TVKTR, and TVKTVr for German, with TVTVR matching the English word, and thus being connected to both words by an edge in our bipartite network (see Figure 1).

Similarly, when computing a modified variant of skip-grams based on n-grams of size 3, where only consonants are taken into account, and in which we allow to replace up to one segment systematically by a gap-symbol (“-”), we can see from Table 1 that the structure of matching n-grams directly reflects the cognate relations, with Greek çeri “hand” opposed to German Hand and English hand (both cognate), as well as Russian [ruka], Polish ręjka (both cognate).

Note that the use of skip-grams here mimics the alignment component of those automatic cognate detection methods in which alignments are used. The difference is that we do not compute the alignments between a sequence pair only, but project each word to a potential (and likewise also restricted) alignment representation. Note also that – even if skip-grams may take some time to compute – our approach presented here is essentially linear in computation time requirements, since the skip-gram calculation represents a constant factor. When searching for potential cognates in our bipartite network, we can say that (A) all connected components correspond to cognate sets, or (B) use some additional algorithm to partition the bipartite network into our putative cognate sets. While computation time will be higher in the latter case, both cases will be drastically faster than existing popular methods for automatic cognate detection, since our bipartite-graph-based approach essentially avoids pairwise word comparisons.

Following these basic ideas, we have developed a new method for fast cognate detection using bipartite networks of sound-class-based skip-grams (BipSkip), implemented as a Python library (see SI 1). The basic work-
ing procedure is extremely straightforward and consists of three stages. In a first stage, a bipartite network of words and their corresponding skip-grams is constructed, with edges drawn between all words and their corresponding skip-grams. In a second, optional stage, the bipartite graph is refined by deleting all skip-gram nodes which are linked to fewer word nodes than a user-defined threshold. In a third stage, the bipartite graph is projected to a monopartite graph and partitioned into cognate sets, either by its connected components, or with help of graph partitioning algorithms such as, e.g., Infomap (Rosvall and Bergstrom, 2008).

Since it is difficult to assess which kinds of skip-grams and which kinds of sound-class systems would yield the most promising results, we conducted an exhaustive parameter training using the data of List (2014, see details reported in SI 2). This resulted in the following parameters used as default for our approach: (1) compute skip grams exclusively from consonant classes, (2) compute skip-grams of length 4, (3) include a gapped version of each word form (allowing for matches with a replacement), (4) use the SCA sound class model (List, 2014), and (5) prune the graph by deleting all skip-gram nodes which link to less than 20% of the median degree of all skip-gram nodes in the data. This setting yielded F-scores of 0.854 (connected components partitioning) and 0.852 (Infomap partitioning) on the training data (using B-Cubes as measure, cf. Amigó et al. 2009 and section 4.2), suggesting that our BipSkip method performs in a manner comparable to the SCA method for automatic cognate detection (List, 2014), which is based on pairwise sequence comparison methods using improved sound class models and alignment techniques. This also means that it clearly outperforms the CCM approach on the training data (scoring 0.8) as well as the computationally rather demanding edit distance approach (scoring 0.814, see List et al. 2017).

### Table 1: Shared skip-grams in words meaning “hand” in Greek, German, English, Russian, and Polish reflect the known cognate relations of the word.

<table>
<thead>
<tr>
<th>IPA Cognacy</th>
<th>çeri</th>
<th>hant</th>
<th>hand</th>
<th>ruka</th>
<th>rębka</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sound Classes</td>
<td>CER</td>
<td>HANT</td>
<td>HENT</td>
<td>RYKA</td>
<td>RENKA</td>
</tr>
<tr>
<td>H-T</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HH-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HNT</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R-K</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 3 F ast Phylogenetic Inference

Methods for Bayesian phylogenetic inference in evolutionary biology and historical linguistics (Yang and Rannala, 1997) are all based on the following Bayes rule:

\[
f(\Psi|X) = \frac{f(X|\Psi)f(\Psi)}{f(X)},
\]

where each state \( \Psi \) is composed of \( \tau \) the tree topology, \( T \) the branch length vector of the tree, and \( \theta \) the substitution model parameters where \( X \) is a binary cognate data matrix where each column codes a cognate set as a binary vector. The posterior distribution \( f(\Psi|X) \) is difficult to calculate analytically since one has to sum over all the possible rooted topologies \( (2L-3)! \) increases factorially with the number of languages in the sample. Therefore, Markov Chain Monte Carlo (MCMC) methods are used to estimate the posterior probability of \( \Psi \).

The Metropolis-Hastings algorithm (a MCMC algorithm) is used to sample the parameters from the posterior distribution. This algorithm constructs a Markov chain by proposing a new state \( \Psi^* \) and then accepting the proposed state \( \Psi^* \) with the probability given in equation 2 where, \( q(.) \) is the proposal distribution.

\[
r = \frac{f(X|\Psi^*)f(\Psi^*)q(\Psi|\Psi^*)}{f(X|\Psi)f(\Psi)q(\Psi^*|\Psi)}
\]

The likelihood of the data to the new parameters is computed using the pruning algorithm (Felsenstein, 2004, 251-255), which is a special case of the variable elimination algorithm (Jordan et al., 2004). We assume that the parameters \( \tau, T, \theta \) are independent of each other. In the above procedure, a Markov chain is run for millions of steps and sampled at regular intervals (called thinning) to reduce autocorrelation between the sampled states. A
problem with the above procedure is that the chain can get stuck in a local maxima when the posterior has multiple peaks. A different approach known as Metropolis-coupled Markov Chain Monte-Carlo methods (MC3) has been applied to phylogenetics to explore the tree space efficiently (Altekar et al., 2004).

3.1 MC3

In the MC3 approach, $n$ chains are run in parallel where $n - 1$ chains are heated by raising the posterior probability to a power $1/T_i$ where $T_i$ is the temperature of $i$th chain defined as $1 + \delta(i - 1)$ where $\delta > 0$. A heated chain ($i > 1$) can explore peaks more efficiently than the cold chain since the posterior density is flattened. The MC3 approach swaps the states between a cold chain and a hot chain at regular intervals using a modified Metropolis-Hastings ratio. This swapping procedure allows the cold chain to explore multiple peaks in the tree space successfully. The MC3 procedure is computationally expensive since it requires multiple CPU cores to run the Markov chains in parallel. As a matter of fact, Rama et al. (2018) employ the MC3 procedure (as implemented in MrBayes; Ronquist et al., 2012) to infer family phylogenetic trees from automatically inferred cognate judgments.

3.2 Simulated Annealing

In this paper, we employ a computationally less intensive and a fast procedure inspired from simulated annealing (Andrieu et al., 2003) to infer the maximum-a-posteriori (MAP) tree. We refer the simulated annealing MCMC as MAPLE (MAP estimation for Language Evolution) in the rest of the paper. In this procedure, the Metropolis-Hastings ratio is computed according to the equation 3. In this equation, the initial temperature $T_0$ is set to a high value and then decreased according to a cooling schedule until $T_i \rightarrow 0$. The final state of the chain is treated as the maximum-a-posteriori (MAP) estimate of the inference procedure. We implement our own tree inference software in Cython which is made available along with the paper.

$$r = \left( \frac{f(X|\Psi^*)}{f(X|\Psi)} \right)^{1/T_i} \frac{q(\Psi|\Psi^*)}{q(\Psi^*)}$$

All our Bayesian analyses use binary datasets with states 0 and 1. We employ the Generalized Time Reversible Model (Yang, 2014, Ch.1) for computing the transition probabilities between individual states (0, 1). The rate variation across cognate sets is modeled using a four category discrete $\Gamma$ distribution (Yang, 1994) which is sampled from a $\Gamma$ distribution with shape parameter $\alpha$.

MCMC moves We employ multiple moves to sample the parameters. For continuous parameters such as branch lengths and shape parameter we use a multiplier move with exponential distribution ($\mu = 1$) as the proposal distribution. In the case of the stationary frequencies, we employ a uniform slider move that randomly selects two states and proposes a new frequency such that the sum of the frequencies of the states does not change. We use two tree moves: Nearest neighbor interchange (NNI) and a specialized Subpruning and Regrafting move that operates on leaf nodes to propose new trees (Lakner et al., 2008).

Cooling Schedule The cooling schedule is very important for the best performance of a simulated annealing algorithm (Andrieu et al., 2003). We experimented with a linear cooling schedule that starts with a high initial temperature $T_0$ and reduces the temperature at iteration $i$ through $T_i = \lambda T_{i-1}$ where $0.85 \leq \lambda \leq 0.96$ (Du and Swamy, 2016). We decrease the value of $T_i$ until $T_i = 10^{-5}$. In this paper, we experiment with reducing the temperature over step size $s$ starting from an initial temperature $T_0$.

4 Evaluation

4.1 Materials

All the data for training and testing was taken from publicly available sources and has further been submitted along with the supplementary material accompanying this paper. For training of the parameters of our BipSkip approach for fast cognate detection, the data by List (2014) was used in the form provided by List et al. (2017). This dataset consists of six subsets each covering a subgroup of a language family of moderate size and time depth (see SI 2). To test the BipSkip method, we used both the test set of List et al. (2017), consist-
Table 2: Datasets (name, concepts, and languages), used for training (a) and testing of BipSkip (b, c) and MAPLE (c). Data in (a) is from List (2014), data in (b) is from List et al. (2017), and data in (c) comes from Rama et al. (2018).

4.3 Implementation

Both methods are implemented in form of Python packages available – along with detailed installation instructions – from the supplemental material accompanying the paper (SI 1 and SI 4). While the BipSkip method for fast cognate detection is implemented in form of a plug-in for the LingPy library and thus accepts the standard wordlist formats used in LingPy as input format, MAPLE reads the data from files encoded in the Nexus format (Maddison et al., 1997).

4.4 Results

Fast Cognate Detection We tested the two variants, of the new BipSkip approach for automatic cognate detection, connected components and Infomap (Rosvall and Bergstrom,
Table 3: Results of the cognate detection experiments. Table (a) presents the results for the performance of the four methods tested on the dataset by List et al. (2017): the CCM method, our new BipSkip methods in two variants (with connected components clusters, labelled CC, and the Infomap clusters, labelled IM), and the SCA method. Table (b) presents the results on the large testset by Rama et al. (2018). The column TIME indicates the time the code needed to run on a Linux machine (Thinkpad X280, i5, 8GB, ArchLinux OS), using the Unix “time” command (reporting the real time value).

As can be seen from the results in Table 3, the BipSkip algorithm clearly outperforms the CCM method in terms of overall accuracy on both datasets. It also comes very close in performance to the SCA method, while at the same time only requiring a small amount of the time required to run the SCA analysis. An obvious weakness of our current BipSkip implementation is the performance on South-East Asian language data. Here, we can see that the exclusion of tones and vowels, dictated by our training procedure, leads to a higher amount of false positives. Unfortunately, this cannot be overcome by simply including tones in the skip-grams, since not all languages in the South-East Asian datasets (Sino-Tibetan and Austro-Asiatic) are tonal, and tone matchings would thus lead to an unwanted clustering of tonal and non-tonal languages in the data, which would contradict certain subgroups in which tone developed only in a few language varieties, such as Tibetan.

The most promising approach to deal consistently with language families such as Sino-Tibetan would therefore be to extend the current approach to identify partial instead of complete cognates (List et al., 2016), given the prominence of processes such as compounding...
or derivation in the history of Sino-Tibetan and its descendants.

Partial cognates, however, do not offer a direct solution to the problem, since we currently lack phylogenetic algorithms that could handle partial cognates (List, 2016), while approaches to convert partial into full cognates usually require to take semantic information into account (Sagart et al., 2019, 10321). In addition to any attempt to improve on BipSkip by enhancing the training of features used for South-East Asian languages, consistent approaches for the transformation of partial into complete cognate sets will have to be developed in the future.

Neither of the two BipSkip approaches can compete with the LexStat-Infomap approach, which yields F-scores of 0.89 on the first test set (see List et al. 2017) and 0.77 on the second test set (see Rama et al. 2018), but this is not surprising, given that neither of the four approaches compared here computes regular sound correspondence information. The obvious drawback of LexStat is its computation time, with more than 30 minutes for the first, and more than two hours for the second test set. While the superior results surely justify its use, the advantage of methods like BipSkip is that they can be used for the purpose of exploratory data analysis or web-based applications.

**Fast Phylogenetic Inference**  We present the results of the phylogenetic experiments in Table 4. Each sub-table shows the setting for $s, T_0$ that yielded the lowest GQD for each cognate detection method. We experimented over a wide range of settings for $s \in \{1, 5, 10, 20, 40, 80, 100\}$ and $T_0 \in \{10, 20, \ldots, 90, 100\}$. We provide the time and the number of generations taken to infer the MAP tree for each cognate inference program and language family. We note that the longest run takes less than fifteen minutes across all the families. In comparison, the results reported by Rama et al. (2018) using MrBayes takes at least four hours on six cores for each of the language family using the SCA method.

We examined which settings of $s/T_0$ give the lowest results and found that low step sizes such as 1 give the lowest results for a wide range of $T_0$. We examined the results across the settings and found that the best results can be achieved with a step size above 20 with initial temperature set to 50. The lowest GQD distances were obtained with the SCA cognates. The BipSkip-IM method emerged as the winner in the case of the Pama-Nyungan language family. The best result for Pama-Nyungan is better than the average GQD obtained through expert cognate judgments reported in Rama et al. (2018). The weakness of the BipSkip methods with respect to the Sino-Tibetan language family is also visible in terms of the GQD distance.

Comparing the results obtained for the SCA cognates obtained with MAPLE against the ones inferred with MrBayes as reported in Rama et al. (2018), it becomes also clear that our method is at least as good as MrBayes, showing better results in Austro-Asiatic, Austronesian, and Pama-Nyungan.

**MAPLE with gold standard cognates** We further tested if gold standard cognates make a difference in the inferred tree quality. We find that the tree quality improves if we employ gold standard cognates to infer the trees. This result supports the research track of developing high quality automated cognate detection systems which can be employed to analyze hitherto less studied language families of the world.

**Convergence** We investigated if the MAPLE algorithm infers trees whose quality improves across the generations by plotting the GQD of the sampled trees against the temperature for all the five best settings of $s/T_0$ (in bold in Table 4) in Figure 2. The figure clearly shows that at high temperature settings, the quality of the trees is low whereas as temperature approaches zero, the tree quality also gets better for all the language families. Moreover, the curves are monotonically decreasing once the temperature is below 12.

5 Conclusion

In this paper we proposed an automated framework for very fast and still highly reliable phylogenetic reconstruction in historical linguistics. Our framework introduces two new methods. The BipSkip approach uses bipartite
Table 4: Results for the MAPLE approach to fast phylogenetic inference for each method. The best step size and initial temperature setting is shown as $s/T_0$. NGens is the number of generations, Time is the time taken to run the inference in number of seconds on a single core Linux machine.

<table>
<thead>
<tr>
<th>Family</th>
<th>$s/T_0$</th>
<th>GQD</th>
<th>NGens</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austro-Asiatic</td>
<td>80/10</td>
<td>0.0155</td>
<td>18080</td>
<td>282.548</td>
</tr>
<tr>
<td>Austronesian</td>
<td>20/80</td>
<td>0.0446</td>
<td>5320</td>
<td>46.698</td>
</tr>
<tr>
<td>Indo-European</td>
<td>20/40</td>
<td>0.0138</td>
<td>5060</td>
<td>46.014</td>
</tr>
<tr>
<td>Pama-Nyungan</td>
<td>40/60</td>
<td>0.1476</td>
<td>10440</td>
<td>224.036</td>
</tr>
<tr>
<td>Sino-Tibetan</td>
<td>80/60</td>
<td>0.0958</td>
<td>20880</td>
<td>295.157</td>
</tr>
</tbody>
</table>

(a) Results for CCM cognates.

<table>
<thead>
<tr>
<th>Family</th>
<th>$s/T_0$</th>
<th>GQD</th>
<th>NGens</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austro-Asiatic</td>
<td>100/90</td>
<td>0.0135</td>
<td>26900</td>
<td>439.005</td>
</tr>
<tr>
<td>Austronesian</td>
<td>100/80</td>
<td>0.0148</td>
<td>26600</td>
<td>285.659</td>
</tr>
<tr>
<td>Indo-European</td>
<td>20/80</td>
<td>0.0211</td>
<td>5320</td>
<td>41.544</td>
</tr>
<tr>
<td>Pama-Nyungan</td>
<td>100/100</td>
<td>0.0722</td>
<td>22600</td>
<td>235.774</td>
</tr>
<tr>
<td>Sino-Tibetan</td>
<td>100/10</td>
<td>0.0138</td>
<td>21680</td>
<td>435.8</td>
</tr>
</tbody>
</table>

(b) Results for SCA cognates.

<table>
<thead>
<tr>
<th>Family</th>
<th>$s/T_0$</th>
<th>GQD</th>
<th>NGens</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austro-Asiatic</td>
<td>40/60</td>
<td>0.0415</td>
<td>10440</td>
<td>151.561</td>
</tr>
<tr>
<td>Austronesian</td>
<td>20/20</td>
<td>0.1022</td>
<td>4780</td>
<td>42.097</td>
</tr>
<tr>
<td>Indo-European</td>
<td>80/10</td>
<td>0.0322</td>
<td>25300</td>
<td>59.023</td>
</tr>
<tr>
<td>Pama-Nyungan</td>
<td>100/40</td>
<td>0.1647</td>
<td>21680</td>
<td>435.8</td>
</tr>
<tr>
<td>Sino-Tibetan</td>
<td>80/20</td>
<td>0.5218</td>
<td>19120</td>
<td>233.173</td>
</tr>
</tbody>
</table>

(c) Results for BipSkip-CC cognates.

<table>
<thead>
<tr>
<th>Family</th>
<th>$s/T_0$</th>
<th>GQD</th>
<th>NGens</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austro-Asiatic</td>
<td>80/80</td>
<td>0.1415</td>
<td>26900</td>
<td>476.113</td>
</tr>
<tr>
<td>Austronesian</td>
<td>40/10</td>
<td>0.0135</td>
<td>26900</td>
<td>439.005</td>
</tr>
<tr>
<td>Indo-European</td>
<td>10/100</td>
<td>0.046</td>
<td>25300</td>
<td>59.023</td>
</tr>
<tr>
<td>Pama-Nyungan</td>
<td>80/70</td>
<td>0.0777</td>
<td>21280</td>
<td>662.447</td>
</tr>
<tr>
<td>Sino-Tibetan</td>
<td>40/50</td>
<td>0.3049</td>
<td>19120</td>
<td>233.173</td>
</tr>
</tbody>
</table>

(d) Results for BipSkip-IM cognates.

We are well aware that our framework is by no means perfect, and that it should be used with a certain amount of care. Our methods are best used for the purpose of exploratory analysis on larger datasets which have so far not yet been thoroughly studied. Here, we believe that the new framework can provide considerable help to future research, specifically also, because it does not require the technical support of high-end clusters.

Both methods can be further improved in multiple ways. Our cognate detection method’s weak performance on South-East Asian languages could be addressed by enabling it to detect partial cognates instead of complete cognates. At the same time, new models, allowing for a consistent handling of multi-state characters and a direct handling of partial cognates, could be added to our fast Bayesian phylogenetic inference approach.

Acknowledgments

We thank the three reviewers for the comments which helped improve the paper. TR took part in the BigMed project (https://bigmed.no/) at University of Oslo when the work was performed. JML’s work was supported by the ERC Starting Grant 715618 “Computer-Assisted Language Comparison” (http://calc.digling.org).
References


A Supplemental Material

The supplemental material was submitted along with this paper and also uploaded to Zenodo (https://doi.org/10.5281/zenodo.3237508). The packages provide all data needed to replicate the analyses, as well as detailed instructions in how to apply the methods. In the paper, we point to the relevant sections in the supplemental material.