Assessing Language Relatedness

Luuk Suurmeijer 2581219
Universität des Saarlandes
luuksuurmeijer@gmail.com

22nd March 2020

For the final assignment of the class Computational Linguistics, I decided to tackle the issue of using computational means to assess language relatedness. In the following sections I briefly prime the reader on traditional language relatedness assessment and contemporary methods of automating this process that are key to the aim of my project: Comparing estimated parameters versus linguistically informed parameters. I discuss the methodology of the present implementation and consequently show and discuss results.

1 On Language Relatedness

Confirming or rejecting the hypothesis that two or more languages are related to one another is traditionally, in historical linguistics, performed using the so called 'comparative method' (Millar & Trask, 2015). The very basic idea is to take hypothesized cognates of languages that are suspected to be related and identify systematic sound correspondences between them. From these correspondences an ancestral state can then be suggested based on historical evidence and information about phonetically likely sound changes. Crucially the proto-state must be one from which the modern words can be derived (a derivation in the form of "sound laws"). An example of a systematic correspondence between English and Dutch is that of /th/ with /d/ respectively (which is hypothesized to come from Proto-Germanic /t/. The comparative method is remarkably robust and has allowed linguists to postulate the ancestor state of most of the European language (Indo-European).

Crucial to this process of verifying hypotheses is the distinction between heritage words and loanwords. Heritage words reflect relatedness whilst loanwords reflect language contact. Loanwords can often be distinguished from heritage words linguistically, because loanwords tend to not adhere to sound laws from the ancestral state. The practice of reconstruction is often messy and data-sparse, blurring the line between loanwords and heritage words to a certain extent. In order to mitigate this effect, reconstruction is mostly done based on a predetermined list of the most stable (least likely to be loaned) words of a language referred to as the "Swadesh list" (containing mostly words for body parts, landmarks and celestial bodies) (Millar & Trask, 2015).

1.1 Automatic Relatedness Assessments

Given the fact that the comparative method is phonetically informed and systematic, one can wonder whether the process could be fully or partially automated using methods from Natural Language Processing. Attempts to tackle this question have been made by numerous authors (Jäger, 2013, Holman et al., 2008) and seem to be based on the notion of "Levenshtein Distance" or "edit distance" between two words.

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & min(i,j) = 0, \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{ai \neq bi} \end{cases} & otherwise; \end{cases}$$

In a nutshell, the Levenshtein algorithm is a table filling algorithm that takes two strings as input and outputs the minimal amount of edits (deletion, insertion or substitution) required to turn the first i characters of a into the first j characters of b. The resulting Levenshtein distance is a measure of string similarity between two words. For example, the Levenshtein distance between the words "honda" and "hyundai" is 3. Because the Levenshtein algorithm gives us a measure of similarity, it is thus intuitively useful for comparing words for relatedness.

The base algorithm has edit costs of 1 for all operations. In many scenarios it could be the case that substituting a [g] for [h] is more costly than substituting ¡g¿ for ¡a¿ (automated spell checking comes to mind). By defining custom edit costs, one can turn the ordinary Levenshtein algorithm into a weighted Levenshtein algorithm (Jurfasky, 2018). This can be achieved simply by adding a resource with all the edit costs between the possible characters that can occur in the string.

Using this method, word pairs of supposedly related languages with a high LD are less likely to be cognates. If all word pairs of the two languages have high LD, the two languages are probably not related. Given that sound correspondences are phonetically informed, it makes sense to use a weighted levenshtein distance for this problem with edit costs based on the likelihood of certain phonetic substitutions, deletions and insertions. For example, we would expect [p] for [b] to be a more likely substitution than [p] for [x].

1.2 Aim of the present project

One could, of course, define such a substitution and deletion/insertion matrix by hand and evaluate cognates based on that. However the nature of the problem begs the question whether these parameters, the phonetic edit costs, can be estimated from data. After all, in related languages, we would expect [b] and [p] to be a more frequent and thus probable correspondence than [b] and [x]. The drawback of this approach is however that defining costs by hand can become tedious if not extremely cumbersome, especially if one decides to take the whole phonological context of a word into account when defining edit costs. Since expert cognate judgements are sparse, a solution to this problem must involve unsupervised learning.

Because of the fact that the current problem involves aligning pairs of words character to character, it is not far fetched to believe methods from the field of Machine Translation could

prove to be useful here. In MT, sentence pairs are aligned on a word by word basis and these alignment (correspondence) probabilities have in the past been succesfully extracted using unsupervised methods such as EM.

The aim of my project is then twofold: To answer the question of whether the language assessment problem can be turned into a machine translation problem and secondly, to see to what extent extracted edit costs outperform phonetically informed handcrafted costs.

In the coming pages, I explicate the methodology of investigation and show and discuss results.

2 Methodology and Parameter Estimation

The project is carried out using version 18 of the the Automated Similarity Judgment Project data base (Søren Wichmann, 2018), a collection of Swadesh lists for more than 5,800 languages which are phonetically transcribed in a uniform way. Only the 40 most stable Swadesh concepts were used in this paper. The database contains roughly 7400 languages. Attested loan words were not excluded. Diacritics in the phonetic transcriptions were ignored.

A number of simplifying assumptions had to be made before continuing the implementation. Firstly, I assumed phonological independence. This means that the substitution costs do not consider the phonological context in which the substitution takes place. Based on traditional analysis, this assumption is demonstrably false. For example, by itself [k] and [s] would not be a likely correspondence, but in the context of front vowels this correspondence is in reality fairly common (in Swedish versus German for example). Secondly, I assumed symmetry between the correspondences, meaning that [k] and [s] is equally costly as [s] and [k] (also false). This also means that I make no distinction between insertions and deletions. I have no information about the conservativity of any of the words in the data. For example, I do not know which form is closer to the original in [broer] versus [bruder] I consider the insertion of [d] equally likely as the deletion of [d].

In order to investigate further, I first implemented a version of the base Levenshtein algorithm, weighted Levenshtein algorithm and a script that reads AJSP text files and generates the average Levenshtein distance between language pairs. I then defined by hand a spreadsheet of all correspondence costs as the logarithm of pseudo-probabilities. I used the log of pseudo-probabilities because it was easier for me to transform phonetic knowledge intuitively this way. Log-probabilities also allowed me to make minimal changes to the Levenshtein algorithm and the log probabilities ended up outputting scores in a similar range as the base levenshtein algorithm.

For parameter estimation, I used an off the shelf implementation of IBM model 2 (fast align). Since IBM models run on two languages, I simply (and naively) enumerated all combinations of 2 from the languages in the data and treated the first element of each tuple as the source language and the second element as the target language. I fed this to the Expectation Maximization algorithm. I tried two training sets, one set included all Bantu languages and the other included all language pairs with an average base Levenshtein distance of below 2 in about 40% of the AJSP data. Unfortunately, both sets ended up with missing entires which rendered it of limited usability (it was usable for tree inference). This is an issue I was unable to solve before the deadline.

When doing pairwise string comparison, I only wanted to take probable cognates into account for my ultimate measure of relatedness (average Levenshtein distance). I ran my algorithm on a number of unrelated languages and took the average distance between those languages as a cognate threshold, which turned out to be around 3.9.

I ultimately used the python packages LingPy and ete3 to infer phylogenetic trees via the Neighbour Adjoining algorithm. I ran the comparisons on a multitude of language families (all of which are included in the AJSP folder), but ultimately tested only on Germanic because of a lack of gold standard phylogenetic trees for linguistic families and I was only comfortable enough creating my own for a subset of the Germanic language family.

3 Results

In figure 1 the inferred phylogenetic trees are compared to the hand-crafted gold standard tree. All trees were above chance in their F-scores, as I compared it with a random tree (not included because clutter). The phonetic parameters outperformed both the base Levenshtein distance and estimated parameter tree. This trend continued in the other languages for which I did not have gold trees.

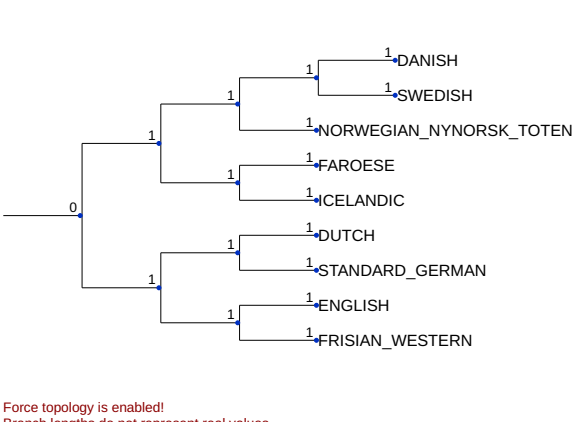
I also examined the distribution of average base pairwise Levenshtein scores between two language families and a sample of random languages from the full AJSP dataset and compared it with the phonetically informed average weighted LD's for the same languages. The distributions are visualized in figure 2. The phonetically informed scores for Germanic are lower than every other score. The Uto-Aztecan scores have higher variability, especially in the lower direction. The random scores have higher variability in the upper region. Phonetically weighted scores seem to outperform base scores by a small margin.

4 Discussion

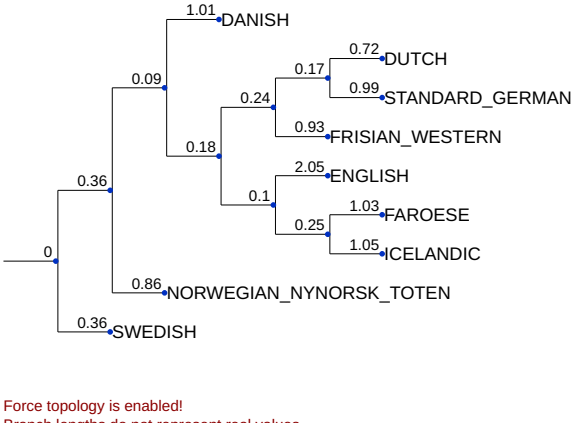
Firstly, it seems it is non-trivial to translate the language assessment problem to a machine translation problem that is completely unsupervised. The EM algorithm got confused by treating multiple languages as the source and multiple others as the target. I suspect a soft-EM approach (that I did not have the time for) might prove more successful: Treat one language as source and all the others as targets and generate translation probabilities and alignments using EM. Then perform MLE on these alignments a number of times to refine the translation probabilities.

In general, I expected the weighted phonetically informed LD's to perform better. It is clear that more refinement is needed to make this problem work. Firstly, there needs to be some way of correcting for inventory similarity. Languages that are unrelated, but have similar phonemic inventories by chance receive higher scores than they should. Secondly the inverse is also true: as seen in figure 2, the weighted phonetic LD's seem to perform worse for language families that are less conservative in their features (like uto-aztecan). Thirdly, refining the simplifying assumptions I explicated should improve performance too. Lastly, training data should consist of multiple language families in order to avoid overtraining towards one language family.

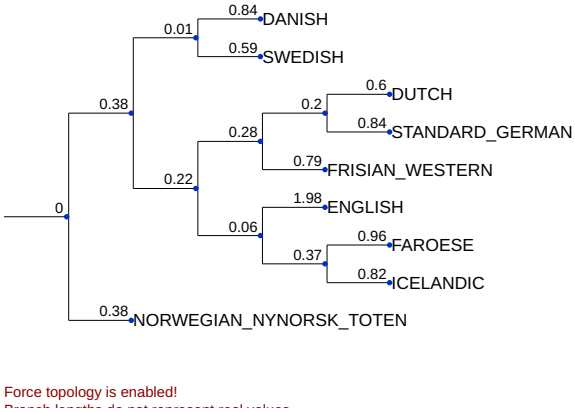
As a personal note, I enjoyed tackling and implementing this issue. However I also realised I



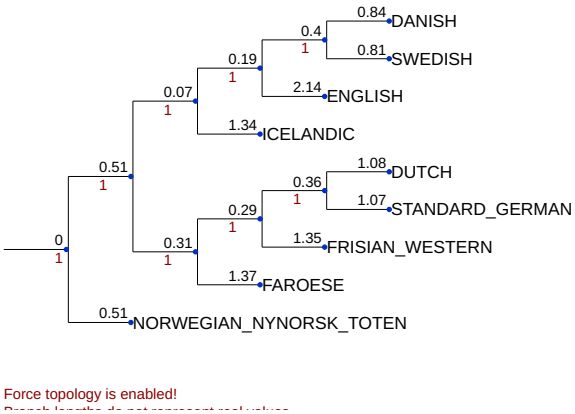
(a) Correct tree



(b) Using average LD, F = 0.29

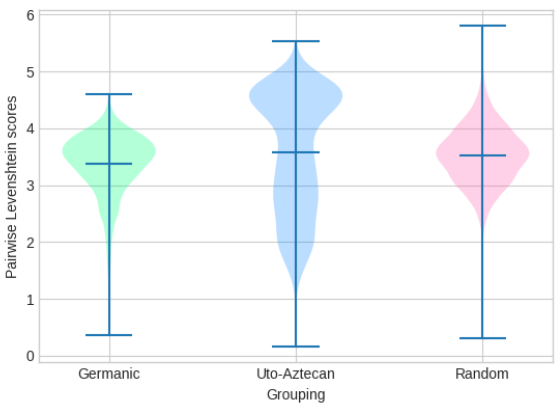


(c) Using average weighted LD, phonetics F = 0.33

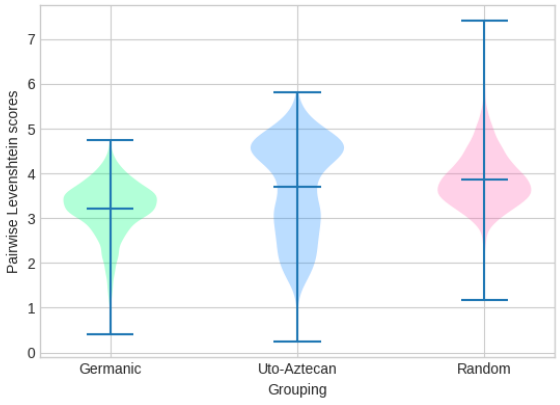


(d) Using average weighted LD, estimated F = 0.29

Figure 1: Phylogenetic trees using different edit costs.



(a) Base Levenshtein Score distribution



(b) Phonetically Weighted Levenshtein Score distribution

Figure 2: Score distributions: Base versus phonetically weighted

bit off way more than I can chew, which resulted in issues with the parameter estimation step. The language assessment problem has many intricacies and gaps, and suffers heavily from sparse data and therefore, despite my passion for traditional historical linguistics, I don't know whether this is the issue for me. I nevertheless learned a lot and hope what I delivered and implemented is sufficient for this class.

References

Holman, E., Wichmann, S., Brown, C., Velupillai, V., Müller, A. & Bakker, D. (2008). Advances in automated language classification. *Arppe, Antti; Kaius Sinnemäki; and Urpo Nikanne: Third Workshop on Quantitative Investigations In Theoretical Linguistics (QITL3), University of Helsinki (2008)*.

Jäger, G. (2013). Phylogenetic inference from word lists using weighted alignment with empirically determined weights. *Language Dynamics and Change*, *3*, 245–291. https://doi.org/10.1163/22105832-13030204

Jurfasky, D. (2018). Minimum edit distance. https://web.stanford.edu/class/cs124/lec/med.pdf

Millar, R. M. & Trask, L. (2015). *Trasks historical linguistics*. Routledge.

Søren Wichmann, C. H. B., Eric W. Holman. (2018). *The ajsp database (version 18)*. %5Curl%7Bhttps://asjp.clld.org/%7D