Language Evolution and Diachrony Generation

Research Project Report

Matthieu Boyer

Laboratoire Lattice

CNRS — ENS-PSL — Université Sorbonne Nouvelle

Under supervision of Mathieu Dehouck

Introduction

Behind any modern language are millenia of languages evolving, through the creation of new words, a change in their phonology or the acception of new meanings for pre-existing words. One issue linguists face when trying to quantify the proximity of two languages is the lack of available data. More precisely, the most precise data available publicly to date on the evolution of languages are the Index Diachronica [ind] for phonology or the $\mathcal{E}$voSem project [FKD$^+$25] for semantics, which are, by essence, very localized on where previous research has been conducted.

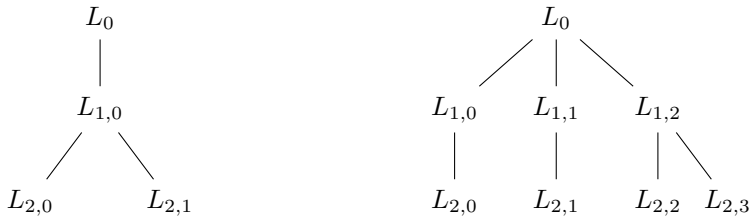
In this project, we thus focused on theorising and implementing a way to generate diachrony to test reconstruction algorithms. The main goal was to have a main frame on which to add successive features, using a modular tree structure which will be described in the next section. The main feature we suggest in this project is to allow a *geographical/sociological* model to be taken into account during the generation procedure, to try and map population dynamics and interactions between languages.

1 The Idea

In this section we will describe the way our generation algorithm works, without considerations of implementation or choice of metrics/random generators.

1.1 Evolution as Random Trees

Consider a language L_0, which we will call our *base language*. We model its evolution through time as a tree. The following examples are both valid evolution trees, with time flowing from top to bottom :



Notice there is no reason for our trees to be binary or even of fixed arity : at any given time, any group of people speaking a language can modify it by independently inventing new words, thus creating as many new languages. A more accurate modelisation would be that each person speaks a different language, but for computational purpose, we have assumed a language can only split, which only models different dialects of the language.

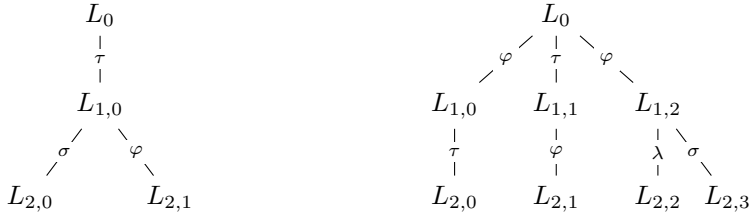
We give ourselves a function Evolve which, given a language l, returns a set S of randomly generated languages that are *close* to l. We will denote by $\text{Tree}\,(r, S)$ for S the tree rooted in r whose subtrees are in S. We will allow a slight notation abuse, by associating an element x of a set of languages to $\text{Tree}\,(x, \varnothing)$. This gives us algorithm 1.

Algorithme 1 One Language Evolution

$leaves \leftarrow \{L_0\}$
$\mathcal{T} \leftarrow \text{Tree}\,(L_0, \varnothing)$
for $n \leq$ `Epochs` **do**
  for $l \in \text{Leaves}\,\mathcal{T}$ **do**
    $S \leftarrow \text{Evolve}\,l$
    $l \leftarrow \text{Tree}\,(l, S)$ ▷ Here $\mathcal{T}$ is modified in place directly, replacing a leaf by its newly generated tree.
return $\mathcal{T}$

Assuming that nodes in a tree represent distinct groups of people, each speaking the language associated to their node, if we ask of Evolve to be easily revertible (that is $\mathbb{P}\,(\text{Evolve}\,(l_1) = l_2) \neq 0 \Leftrightarrow \mathbb{P}\,(\text{Evolve}\,(l_2) = l_1) \neq 0$) we see there is no need for algorithm to ever resolve splits (and we will see later that we do not want to).

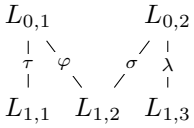
Let us specify a bit what Evolve will do. We would like to be able to model evolution in either phonology (φ), semantics (σ), syntax (τ), or lexicon (λ), which we will call *evolution types*. One way to do so would be to pre-generate the full tree then specify the way the languages will evolve and finally compute the evolution of languages accordingly. However, since we would like to take the proximity of two languages into account (we will explain how in the next section), we need the proximity of two languages to compute the evolution type, and thus we need an algorithm which acts directly on the last level of nodes (which map the *actual time* languages). Even though Evolve only acts on one language, since we want it to run in the same time step as our multi-language evolution functions, we still need it to verify this property. In this sense, we want Evolve to be an abstract construct, which randomly selects an evolution type, and then computes the random evolution on our language. We then have the following type of trees :



This does not modify our algorithm, but simply our tree structures since our edges are now labeled.

1.2 The "*Collision*" Hypothesis

We now make the assumption that interactions between languages cause evolution in a way different of spontaneous evolution, including features of both languages. We then represent our trees in the following way :



A node having two ancestors thus means that it is the result of the collision of the two languages. There is, in principle, no necessity for the labels of the two parent edges to be of the same type. This means that languages $L_{0,1}$ and $L_{0,2}$ have been sufficiently in contact so that speakers of $L_{0,1}$ have adapted a variant of $L_{0,2}$ and reciprocically.

This allows us to evolve our algorithm a bit, as shown in 2.

Algorithme 2 Two Language Evolution

Stack $\leftarrow \varnothing$
$\mathcal{T} \leftarrow \text{Tree}\,(L_0, \varnothing)$
for $n \leq$ `Epochs` **do**
  for $l \in \text{Leaves}\,\mathcal{T}$ **do**
    $S \leftarrow \text{Evolve}\,l$
    Push (Stack, $l \leftarrow l \cup \text{Tree}\,(l, S)$) ▷ Here $\mathcal{T}$ is not modified in place, we just add the operations to be done to a stack.
  for $l \in \text{Leaves}\,\mathcal{T}$ **do**
    $l^\dagger \leftarrow \mathcal{P}_l\,(\text{Leaves}\,\mathcal{T})$ ▷ Choose another leave according to a probability distribution $\mathcal{P}_l$ to be defined.
    $S \leftarrow \text{Collision}\,\left(l, l^\dagger\right)$
    Push (Stack, $l \leftarrow l \cup \text{Tree}\,(l, S)$)
  Apply(Stack) ▷ We modify $\mathcal{T}$ at this step, by applying without destruction all the previous steps.
return $\mathcal{T}$

Here we consider two steps of generation : one for spontaneous evolution, the other for interaction-driven evolutions. Both are to be randomly computed for each leaf, allowing for more parameters to be taken into account and a better modelization. Discussion around the probability distribution to take will be done in the following section. Let us simply

discuss the properties of collisions in Collision. Collision should, in the same fashion as Evolve, provide a way to choose the type of collision, for each parent. Moreover, Collision should take into account the *linguistic* proximity of the two considered parents when computing the resultant language, as switching from a language to a close one requires less effort than changing from a language to farther one. Furthermore, there should be a way to discuss the *strength* of the interaction, which will be linked to the way we decide to compute the probability distribution $\mathcal{P}_l$ for each language, since more probable collisions should also be stronger. Indeed, remember our probability distribution serves to map which languages are more in contact with each other and thus speakers are even more incline to change a part of their language to be understood more easily by speakers of the other language. Of course, we also ask of Collision to be *easily revertible*, in the same sense as for Evolve.

1.3 Collision Probability : A Manifold of Languages in Life, the Universe and Everything

We will now take a look at the way we might define the probability distribution on languages. Remember that $\mathcal{P}_l$ is supposed to model the probability a language interacts with l. The main idea we developed is the following : we define a *geographical* embedding for each language, associating to any language a point in a certain space, with a certain geometry. This gives a distance metric for each and every language, which may depend on the language. In a sense, we create a manifold for all our languages, and the geodesic metric of that manifold is our comparison metric. Then, we decide that $\mathcal{P}_l$ is a distribution proportional to 1 over $d_l(x)$ renormalized.

For the choice of the manifold, we will limit ourselves to choosing the distance metric, as defining a manifold with the right shape is often too difficult. As examples, the first three manifold we have chosen to test are :

- The simplex, that is, $d_l(x) = 1$ for all l, x.
- The euclidean space $\mathbb{R}^3$, that is we map all our languages to some point in $3D$-space and compute the ℓ^2 distance between them.
- The 2-sphere $\mathbb{S}^2$ where each language is associated to a pair λ, φ of latitude and longitude, representing their position on a globe, then :

$$d_{(\theta_1,\lambda_1)}(\theta_2, \lambda_2) = \arccos\left(\sin(\varphi_1)\sin(\varphi_2) + \cos(\varphi_1)\cos(\varphi_2)\cos(\lambda_2 - \lambda_1)\right)$$

The ideas is that each language is associated to a point representing real-life features independent of the linguistic features of the language, such as geographical position, surface over which the language is spoken...The metric then represents the ease with which speakers of both languages can talk to each other, and can represent geographical proximity or overlap, geographical barriers, social and cultural proximity...While we talk about this function as a distance, it is important to note it is more of an *outer* metric, independent of the language features.

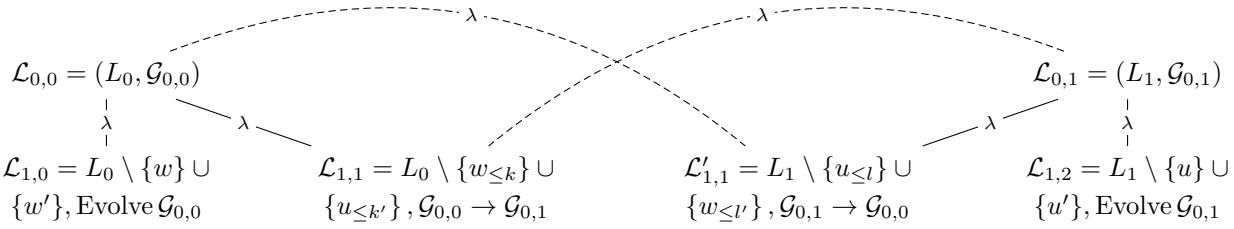
For the implementation, we will suppose a language can only interact directly with languages alive at the same epoch/reference time as itself. While this may seem obvious, one could think about some groups of people using antique languages between themselves, being influenced by cultural references. A way to avoid this complication could be to add a very important weight to the epoch of a language when computing $\mathcal{P}_l$. Geometrically this is the same as adding a dimension to our manifold and computing $\mathcal{P}_{l,t_0} = \mathcal{P}_{l,t\leq t_0} \times U$ where U is a probability distribution used to renormalize our distribution, allowing for different weights to different epochs. Furthermore, especially in this setting but also in general there is no reason that our metric is actually a metric. It's only requirement is for it to be well-defined and positive. We will still employ the term *metric* for our function independently.

2 Results and Troubles

2.1 Current Implementation

At this point, most of the features connected to the *connection* hypothesis are implemented, with Algorithm 2 fully implemented, and all the *geographical* metrics detailed in Section 1.3 being implemented.

The code is structured in the same way we described the algorithm : a language is an abstract object comprised of a *linguistical* observable and a *geographical*[1] observable, both having their own Evolve method. However, since there is very few evolution data available in most domains, we only implemented a *linguistical* observable based on a lexicon of words that spreads across all the children of a certain language.



The way our lexicon works is by creating a set of words for each of our *base* languages $(\mathcal{L}_{0,i})_i$ and then evolving it. $\text{Evolve}\,(\mathcal{L} = (L, \mathcal{G})) = L \setminus \{w\} \cup \{w'\}, \text{Evolve}\, \mathcal{G}$ then works by removing a few words from the language and adding a few new ones from no other language, and randomly evolving the spatial part. Here, $\mathcal{L}_{1,1}$ is not the complement to $\mathcal{L}'_{1,1}$. For collisions, $\to$ designates the spatial evolution, which is dependent on the type of representation chosen, and $L_0 \to L_1$ is defined by taking a randomly generated set of words from L_1 and adding it to L_0 from which we have removed a randomly generated set of words.

The last thing we need to define is the random distribution used. We have already discussed geographical proximity and the way we use it to choose a *neighbouring* language. To decide whether or not we actually create a collision/a random evolution or if we simply keep our language for one more epoch, we set thresholds $1-\alpha$ representing our random evolution probability, $1-\beta$ representing our collision generation probability. Moreover, we try limiting ourselves to a certain maximum width per epoch in the trees, to try and limit our total number of nodes. If both fail, we pass our language through, as if it were completely isolated and unchanged.

Our problem is then to find *optimal* parameters $\alpha, \beta, \text{Evolve}, \text{Collision}$ and distribution $\mathcal{P}$.

2.2 Difficulties with Performance Checking

The main issue with our method is not implementation, it is the lack of ways to compare our method with different parameters to perfect it. Before anything, we need to confirm our hypothesis on collisions, and show that adding collisions according to a certain metric may lead to better results.

We thus need a way to prove that, on a localized geographical space, our method using collisions allows for better results. We then need to find a way to compare two languages together/the result of the evolution from two families. To do so, and try to connect our results to observed data, we have used [FKD+25] as a lexical bank. We chose for our base languages two proto-families, Germanic and Indo-European, whose words have in majority [2] derived semantically to words in modern French, German, English, Dutch, Spanish, Italian and Danish. Note that one could do the operation in the other way, choosing for base languages modern ones and going all the way back to families who have had at least one word generated, from the *reversibility* property we ask of Evolve and Collision. Here, we do not take phonetical/orthographical/semantic changes, and just consider for our observable words in the proto-families and the

1. To understand as "*tied to real-world interactions*".
2. More than 50% of words from those families are concerned.

percentage of dialexification between two modern languages. This observable is easily computable for our evolution trees as we know the full list of words in all our base languages and the list of words in all our computed modern languages. We simply regroup a few languages under the same name in the last step to account for all the splits we have previously done.

The first major issue lies in the difficulty in choosing a metric which makes sense for our result. Since our algorithm generates a random number of languages, and we cannot regroup them only basing ourselves on geographical consideration [3], the best way we have found was to randomly sample just the number of languages that we have data on and compare the resulting dialexification probability matrix to the observed one, using the ℓ^2 metric. We cannot realistically compute this loss for all subsets of languages of the right size, since in our case the size is 7 and that would necessitate around $\binom{7}{3^d} = \mathcal{O}(3^{7d})$ computations. This leads to a first estimate of how far apart can languages drift, though this estimate can be improved before we show that the interactions observable are actually useful to get better results.

However, here lies the second major obstacle to checking the performance of our method : the time complexity. The analysis provided below will be done in the worst case, to avoid probabilistic considerations. Our algorithms creates sub languages, which in turn can create sub languages : at each epoch, we can increase the number of nodes in our trees by a factor of 3 and we do so in quadratic time in the number of nodes at each epoch (we need to compute all the spatial distances between each pair of languages present in the same epoch) if we consider each random evolution to be done in constant time [4] and if we consider computation of distances in constant time [5]. This means if we have d epochs and k base languages, the algorithmic complexity for the evolution step scales in $\mathcal{O}(3^{2d}k)$ and this scales poorly as we add more words to our base lexicons. Moreover, the sheer number of random factors in the result of our generation algorithm (as well as our measure of accuracy/realism) necessitates that we do multiple reruns for each test of our parameters. We are also forced by our accuracy computation to simply try and find results for lots of parameters instead of trying to minimize our function, as it is nor convex nor actually deterministic.

2.3 The Results

In the following plots are, in the x-axis, the values of the α parameter, and in the y-axis the values of the β parameter (so the co-probabilities of generating evolution). Instead of directly plotting the losses for each pair of parameters, we decide to plot methods against method using a color map : in color (the scale is on the right of each plot and depends on the plot) is the difference of the mean losses computed as explained in the previous section.

The computations below are done for evenly-spaced α and β values between 0 and $1 - \varepsilon$ included. We have taken a step of 1/16 and done 7 runs for each choice of parameters. The algorithm is run for 7 epochs each time. For the computation using the spherical geometry, it took 47 hours to compute the matrices.

This gives us a first plot in Figure 1 comparing the results obtained when using euclidean geometry with base languages $\left(\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}\right)$. Here cube refers to the fact our base lanuages are placed in the vertices of a cube. For the comparison with the sphere however, we have really different results, as seen in Figures 2 and 3

3. Without adding a large bias towards our hypothesis.
4. Which is already dubious for our representation of lexicons as sets/hashmaps of strings, and should not be considered true in general, as phonetics for example are consistent across all words in a language.
5. Which might not be always true, especially when considering the epoch as an additional dimension.

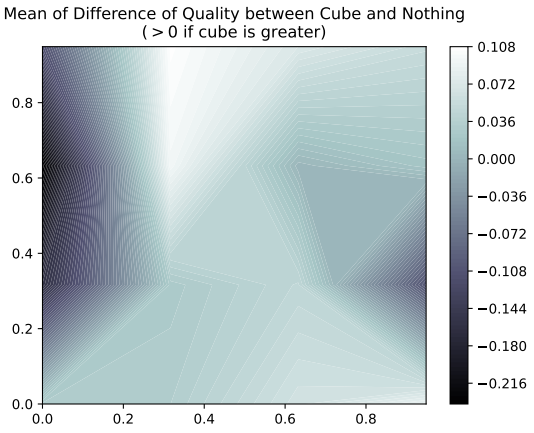


FIGURE 1 – Plot of computed differences in accuracy between the euclidean space geography and no geography.

Here we see for example that the difference of accuracy is slightly in favor of the cube when β is small or α is around .5. This might come from the assimilation of our metric with the constant metric when space is overcrowded, i.e. for $\alpha \to 0$. For large values of α, the issue is the same but opposite : since there are few new neighbours, the distance metric is still essentially the same, and β acts as a second α. Remember that there is a large variance in our results (around $\sqrt{2.10^{-2}}$, and that since .1 in difference is attained in both directions, it might not be really significative.

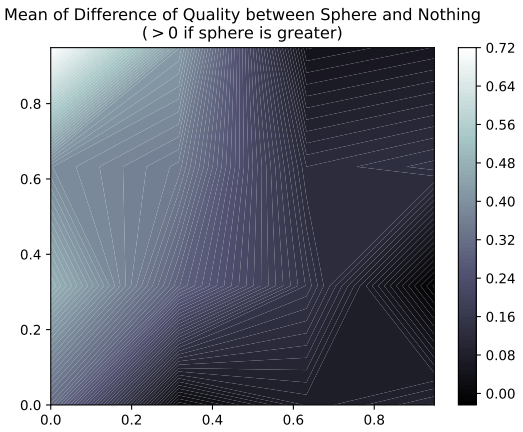


FIGURE 2 – Plot of computed differences in accuracy between the $\mathbb{S}^1$ sphere and no geography.

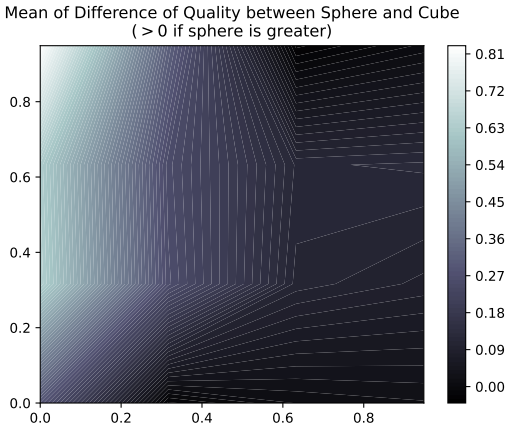


FIGURE 3 – Plot of computed differences in accuracy between the euclidean space geography and the $\mathbb{S}^1$ sphere.

In Figures 2 and 3, the variance problem we had is not relevant since, while precision is still certainly off, our values are almost always positive and with magnitudes far greater than in Figure 1. This shows that our method, while not significantly proven better, allows for better results with objective our reality when using the geodesic metric on $\mathbb{S}^1$ as a collision generator.

Références

[FKD+25] Alexandre François, Siva Kalyan, Mathieu Dehouck, Martial Pastor, and David Kletz. Evosem : A database of dialexification across language families. Online database., 2025.

[ind] Index diachronica. `https://chridd.nfshost.com/diachronica/`.