VRIJE
UNIVERSITEIT
BRUSSEL

Master thesis submitted in partial fulfilment of the requirements for the degree of
Master of science in de ingenieurswetenschappen: Computerwetenschappen

USING COALESCENT SIMULATIONS TO TEST HYPOTHESES OF LANGUAGE CHANGE

Master Thesis Computer Science

Lucas Conchuela Nogales

Academic Year 2020 - 2021

Promotor: Prof. Dr. Bart de Boer
Advisor: Peter Dekker

Science and Bio-Engineering Sciences

VRIJE
UNIVERSITEIT
BRUSSEL

Proefschrift ingediend met het oog op het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen

TOEPASSEN VAN COALESCENTIESIMULATIES OM HYPOTHESES OVER TAALVERANDERING TE TESTEN

Master Thesis Computerwetenscappen

Lucas Conchuela Nogales

Academiejaar 2020 - 2021

Promotor: Prof. Dr. Bart de Boer
Advisor: Peter Dekker

Wetenschappen en Bio-ingenieurswetenschappen

Abstract

The coalescent has been used in population genetics for a long time and has recently seen more use in the field of linguistics, particularly on genetic data in gene-language coevolution studies. We developed a novel method to use the coalescent theory on solely linguistic data in order to infer the exchange and ancestry between languages. We created a model that relies on the comparison of linguistic data by using summary statistics of the linguistic data in order to infer inputs that can be used for analysis. The summary statistics use the data in order to understand what the relation between different languages is. We will look at how different languages are by looking at how much difference in cognates there are. The inputs for our analysis are the population splits and the borrowings from one language to another. Inputs of the model are optimized in two steps in order to get the best possible representation of the languages by the model. We first find a good starting values for our model and then try to optimize these values by taking small steps to the best possible inputs for our model. By aggregating the results that the model finds for Indo-European languages, we have found a good accuracy in the clusters found and the ancestry that was built. We have also shown that the program works for smaller sets of languages, requiring only 208 concepts from the Swadesh list per language and without relying on genetic data. We believe that the use of the coalescent theory on linguistic data is promising, because it allows the reconstruction of a complex history and only requires contemporary linguistic data.

Contents

Chapter 1

Introduction

How languages are related to one another has been attributed to many factors. A language can be similar due to common ancestry, in the form of a common ancestor language, chance, because some properties are more common, for example having verbs and nouns or consonants and vowels, or borrowing, due to higher amount of potential exchange or borrowing from geographically closer neighbors (Daumé III, 2009). In this research, we will explore two of those factors. The first factor, which is the one where we expect languages to be more similar because of a common ancestor, and the third factor, which expects languages to share some features with each other. For the latter, we are more interested in the exchanges due to borrowing, independent of geographical distances. To do this, we will use a model from bioinformatics called the coalescent theory for linguistic phylogeny and language evolution analysis. The coalescent theory is usually used for population genetics on gene data, but we want to evaluate this technique in order to find out if it can be used in (computational) linguistics. The coalescent theory can be a useful tool of analysis for variation in DNA data, which has been beneficial in research for "pharmacogenomics, animal and plant breeding, conservation genetics, epidemiology genetics, medicine and forensics" (Rozas et al., 2003). The reason that the coalescent theory is used in research is that it provides a method that can evaluate hypotheses by working backwards in time, starting from the sampling time. This means that there is no requirement for historical data.

We want to expand on this list of topics where the coalescent theory can be useful by applying it to linguistics. We want to understand how languages are related to each other in order to better grasp how to reconstruct a language family tree and how to find exchanges between languages without using historical data as a basis. Linguists often use the comparative method from comparative linguistics in order to form their conclusion on how languages are historically related. Discrete lexical, morphological and phonological data is used as a basis for comparative linguistics (Gray & Atkinson, 2003). We provide an alternative that is based on a more statistical and computational approach to solve this problem, compared to traditional historical linguistics that can be found in Campbell (2013). Our method provides a way to detect borrowing and reconstruct the ancestry between languages with a small amount of linguistic data.

Aim of Work & Motivation In this work, we want to graph the vertical and horizontal exchange between language. The result would thus be in two parts: one graph that shows the vertical exchange, which will try to organize languages in a way to show the ancestry of the languages, grouping them in language groups or a family. The second part would be interested in the horizontal exchange, mostly happening due to exchange between languages.

Our model will use contemporary linguistic data in order to deduce what change has had to

happen in the past to fit this linguistic data. We start from a linguistic database of word lists (Dunn, 2012) and present a method to extract knowledge from this data, loosely based on the work done in Thouzeau et al. (2017).

The problem we have to solve with this work is the problem of finding a good way to integrate a technique from biology into a linguistic framework. If done successfully, this could mean that we can use the coalescent theory as an accurate framework for work in linguistics. For example we could evaluate different hypotheses and find out which one of the hypotheses has more support. This would be helpful because the coalescent requires no historical linguistic data. The only data that is required is contemporary data that can be gathered in a more easy way.

Once the data has been gathered, a hypothesis could be built that describes a scenario. A hypothesis would then consist of how languages are related to each other. This includes dates for the most recent common ancestor and how much exchange has happened between the languages for example. These are inputs to a model that is able to find if accept or reject a certain hypothesis using summary statistics as a tool for evaluation.

This means we also require a way to validate the work that has been done to build this bridge between computational linguistics and bioinformatics. The validation for vertical exchange can evaluated with our current knowledge by comparing the results to the known language tree of the languages inside the hypothesis. For horizontal exchange, it is more complex as there is not one specific that metric encompasses all the borrowing between two languages. For example mutual intelligibility can not tell the complete story of borrowing. To evaluate borrowing, we have to look at specific case studies of certain languages to see whether or not the technique accurately represents borrowing in known cases. If this is true, we can generalize it to case studies where we do not know about specific borrowing.

Structure This document starts off with a description of the current literature on the topic of computational linguistics concerning language change. We then compare different methods to look where they differ or resemble each other. We look into problems of comparative methods to reconstruct language change. This will help us build a methodology than can be comparable to the state of the art methods. We then go in deeper details about our method to find the best model, given the linguistic data that is given, explaining in detail every choice that was made in the methodology. We also explain what linguistic data has been chosen and also give some insight on the key concepts to understand the method we used to build the language trees.

We then conclude with examples of the methodology applied on multiple different language families, discussing further what the findings are. We conclude with a review of what has been learned and give a summary of the work.

Chapter 2

Background

The coalescent theory is at the center of computational population genetics (Hudson et al., 1990). The mathematical foundation of the coalescent theory is a continuous-time Markov process that has been studied in Kingman (1982). The biological insight that was developed from this was that this theory could be used to model population genetics. The coalescent theory starts at the time of sampling and goes back in time by coalescing lineages in their most recent common ancestor. This means that we will group up individuals until we find a most recent common ancestor. This method of working backwards in time is usually easier to work with, because it only requires data that is available today instead of historical data. In other words, the goal of the coalescent is then to try to find a past that explains the observed variation of samples in the present.

Rosenberg & Nordborg (2002) shows how the coalescent can be used to infer the most likely scenario of human migration, which cannot be done with the phylogenetic methods, because they cannot estimate the mitochondrial DNA tree unambiguously. The coalescent differs from this because we can evaluate the likelihood that a model gives us the genetic data that we know of now. The coalescent also allows us to model population of individuals, which makes it possible to model migration in a formal way. This unique property is useful because it allows to build a more complete model that can be evaluated statistically.

Many different aspects can come into play when modeling a population genetics with the coalescent theory. For this reason, a lot of work has been done on parameters that affect a model. Population size (Kuhner et al., 1998), population structure (Beerli & Felsenstein, 2001), migration (Beerli & Felsenstein, 2001), recombination (Kuhner et al., 2000; Griffiths & Marjoram, 1996), selection (Hudson & Kaplan, 1988), mutations (Hudson et al., 1990; Drummond et al., 2002) are some examples of ways to influence the likelihood that an allele is passed down from generation to generation (Nordborg, 2004).

Most of the applications of the coalescent theory will be found in the field of bioinformatics, especially on the topic of population genetics. Studies on ecology (Morlon et al., 2010), discovery of species and biodiversity (Monaghan et al., 2009), evolutionary biology (Liu et al., 2010), epidemiology (Pybus et al., 2001) and many more have used the coalescent theory as a basis for their approach. It can safely be said that disciplines related to biology have been using this technique for decades. This technique has also successfully been applied in fields closer to linguistics in recent years by analyzing populations that speak the same language. Daumé III (2009) has modeled the linguistic phylogeny using the coalescent as a probabilistic model of trees. He compared the model found by the coalescent without areal knowledge with the model that used areal knowledge discovered thanks to a Pitman-Yor process. Zlojutro et al. (2009)

created 25 demographic scenarios and used the coalescent to find the founding population for a specific Turkic-speaking population called the Yakuts. Thouzeau et al. (2017) analyzed a wide set of linguistic and genetic data of Central Asia and found several differences between linguistic and genetic histories. This gives us a better understanding of the difference between these two histories.

In the field of linguistic, the coalescent provides a statistical method to allow comparison of models of demographic histories.

As shown, techniques from the population genetics field have previously been applied in linguistics. Research has even shown that there is not only a link between the disciplines, but also between genetics and linguistics. Atkinson & Gray (2005) review parallels between processes of biological and linguistic evolution, as observed by Darwin (2009), even in the research methods used and problems the two disciplines try to solve. Sokal et al. (1990) created a model to show the genetic differences and similarities between groups of speakers in Europe, which shows that it is very likely that there is a strong link between genetic diversity and language diversity. Hunley et al. (2008) investigated the Northern Island Melanesia region and found that linguistic features have been less likely to diffuse across population boundaries than genes. In New Britain, they found a very strong correlation between genetic, linguistic and geographic distances. Although research has found many links between genes and language, the way they are transmitted from one person to another is very different. Genes are transmitted from the parents, while language is learned from anyone that can speak the language. Gong (2010) talks about three types of language transmission: horizontal, vertical and a combination of both (oblique) language transmission. In this definition, horizontal means from peers in the same generation, vertical means from a biologically related member of a earlier generation and oblique means from unrelated members of a earlier generation.

Ranacher et al. (2021) also mentions a similar concept of horizontal transfer. Horizontal transfer happens when a language introduces words or structural features from another language. This is not restricted to loan words, it can also happen over a long period of time. In this case, we are not restricted to a certain generation. To look for horizontal transfer, Ranacher et al. (2021) uses Bayesian clustering methods to find contact areas where this transfer has happened or still happens. The vertical transfer happens through an evolutionary process, where each generation passes the language with some variation on to the next. This does not mean that it necessarily involves parents, unlike genetic transmission. Combining this with the previous knowledge, we can generally define horizontal transmission as language contact (between speakers of the same or a different language). Vertical transmission is more concerned with the ancestry or shape of the language family or group. This type of transmission models what language features are kept or not over time.

Some computational methods have already been applied to detect borrowings. For instance, Nelson-Sathi et al. (2011) used 2346 etymologically related words from 84 Indo-European languages and correctly identified 94% of the known borrowed words in the data by analyzing the community structure in cognate networks. List, Nelson-Sathi, et al. (2014) used a minimal lateral network (MLN) approach originally designed to study lateral gene transfer in order to detect potential transfer events in etymologically related words. In List, Shijulal, et al. (2014), they also use a network approach instead of a tree model to test the vertical and horizontal aspects of language history in Chinese dialects which cannot appropriately by a tree model. Willems et al. (2016) adapted hybrid detection algorithms from genetics to study the hybridization of Indo-European language family. Jacques & List (2019) argues that although tree structures do not give a complete representation of language history, they should not be completely replaced by network approach.

Researchers have previously tried working with genetic data to explain the evolution of lan-

guages. The genetic data has helped reconstruct complex demographic history of human populations. Palstra et al. (2015) uses the coalescent to reconstruct movement of populations in Central Asia by testing multiple hypotheses of gene flow. If we can find evidence for coevolution of language and genes, language evolution could also help understand human evolution better. Atkinson (2011) used phonemic data and found that there is a decrease in phonemic diversity the further the distance is from Africa, claiming that is shows parallel mechanisms shaping genetic and linguistic diversity. Other research was built on this idea, but this time, using genetic data to test this hypothesis. Creanza et al. (2015) used phoneme inventories from 2082 languages and microsatellite polymorphisms, which are repetitive DNA sequences of several base pairs, from 246 populations and found that phonemes are not subject to drift in the same way as genes. Relatively isolated languages had more variance in the phonemes than languages with many neighbors. Many other researchers have tried exploring the combination of genetic data and linguistic data. Verdu et al. (2017) used genetic data together with linguistic data and found coevolution of genetic and linguistic variation inside the creole-speaking population of Cape Verde. Thouzeau et al. (2017) used microsatellite data from populations and phonetic transcriptions of words from the extended Swadesh list (Swadesh, 1955) for these populations. They show how some linguistic exchanges happened without genetic exchange in Central Asia using the coalescent. Hunley & Long (2005) found little evidence for correlation between linguistic and genetic diversity, comparing distances from 1056 mtDNA sequences of Native North American populations with glottochronological distances.

In our research, we want to build the most likely language family trees, together with the expected linguistic exchange, using only linguistic data. To model this, we use the coalescent, which was previously used in research with genetic data. We will use cognate data for our method.

A (lexical) cognate is defined as “a language or a linguistic form which is historically derived from the same source as another language/form” (Crystal, 2008). The source of two cognates words is the same word. For example, the French word ”père” and the Spanish word ”padre” are cognates, because they both have the same source (word). It might seem as an easy task to detect cognates: two words that look like each other are cognates. Although this applies for some words, this is definitely not the always the case. Many words that look the same come from different sources. This phenomenon is called ”false cognates” (Moss, 1992).

Determining cognates requires a difficult procedure by historical linguists, called the comparative method, which involves comparing many forms for many different languages. Because of the difficulty of this task, researchers have looked at techniques to automatically find real cognates. Some of the methods are: Normalized Edit Distance (Levenshtein, 1966), Online PMI (Rama et al., 2017) or Sound-Class Based Alignment (List, 2014). We have not used any techniques to find cognates as the data we used already provided manual, expert, linguist cognate judgments. Rama et al. (2018) explains that these automated methods for cognate set predictions provide results that are very close to reality, although there are some improvements possible.

Other research has been done with more focus on historical linguistics in an attempt to reconstruct the history of a language or group of languages. Bouckaert et al. (2012) evaluated out two hypotheses for the origin of the Indo-European language family. They created a phylogenetic reconstruction algorithm and using cognates from 103 languages of this family, they were able to favor one hypothesis using an extended Bayesian phylogeographic inference framework. This helped infer the geographic origin of the Indo-European language family. This method is a Bayesian Markov chain Monte Carlo method, which is a different method than the coalescent that was previously explained.

Chapter 3

Methodology

3.1 Method Description

Before we start explaining the details about the method, it is important to have a more general understanding of what steps are taken to achieve the results we aimed for in the introduction. First, linguistic data will be used to get summary statistics of the data. Using the coalescent, we will create a model and calculate summary statistics from the output of this model. Summary statistics are measures to provide information about the data. We will iteratively improve the model by comparing both summary statistics until the model corresponds as well as possible with what we have found in real-world data. Multiple models of different sets of languages can be grouped to better understand linguistic exchange and the structure of language families. This will make it possible to compute horizontal and vertical exchange of a certain language group or family.

Data Collection In the first step, we need to collect linguistic data about the languages or linguistic varieties we want to research. In our method, we are going to heavily rely on cognates, because they provide an easy way to compare languages on some basic concepts. For this purpose, we use an existing database called IELex (Dunn, 2012). Databases that contain only word forms cannot be immediately be used, because word forms can be deceiving when compared to each other. This is because word forms that look similar are not always cognates. There are steps that can be taken to transform a word form database to get cognate data by using automatic cognate prediction. IELex already provides us with cognates, which means that we have to do no preprocessing work to get something useful out of the data.

Forming Hypotheses The next step is to design a basic tree structure that can be modeled with the coalescent theory to represent the languages or linguistic varieties in a simplified manner. This structure should be able to be used to form an analysis of the vertical end horizontal exchange between the languages. Each tree will represent a hypothesis with multiple inputs. We need to create a method that takes user defined parameters as input, such as population splits or mutation rate, and returns data that can be compared to real-world data by means of summary statistics. Ideally, our hypothesis should contain every language that we want to analyze. However, this is problematic, due to the fact that the number of inputs to the model increase very quickly. This is why we prefer to create small basic trees that we later on combine during the analysis. The most basic structure that can be chosen is a tree with three leaves, each one representing a language. This tree contains two parent nodes, one that is the parent of two

of the languages and another that is the parent of the previous node and a third language. We chose three languages because it is the smallest number of languages that gives useful information and can be reused. This tree structure will be used to model the coalescent simulation. In other words, the structure of the tree is fixed and we can provide a hypothesis by giving the parameters a value.

Running Simulations Now that we understand the basic structure and the inputs that go into a model, we can start simulating trees. We can simulate a tree by giving the tree a structure and giving values to parameters, such as population splits and migrations. This is our hypothesis. One run of a simulation results in a most recent common ancestor, together with a number of mutations. We are able to use this information as if it was linguistic data.

Summary Statistics & Optimization Once we are able to run coalescent simulations, we have to compute some summary statistics that will be helpful to understand the relation between the three languages. We can compute them from the output of the coalescent simulation we have gotten from the previous step. These summary statistics provide us a way to reliably accept or reject a certain hypothesis that is modeled by the inputs.

The most important requirement for the summary statistics is that they should be able to be computed from the outputs of a coalescent simulation. The specific representation of the output of the coalescent simulations will be explained in the section about the language representation using the coalescent theory.

Included in this step is the procedure that is concerned with the optimization of the inputs as a way to make the summary statistics of the simulations be more similar to those of the real-world data. The result of this step will be a combination of three languages together with the inputs (population split and exchange) that give the most accurate model.

An alternative optimization method based on an ensemble learning regressor will also be given to speed up the process of optimization. The first method will be used for more precise results, while the second method will be used for faster results.

Analysis of Hypotheses The last step will be an analysis of what can be found if we aggregate the output data after the optimizations of the previous step. This is where we will combine the different smaller trees that we have computed into a larger tree. We will combine the inputs of two languages by using the average of these inputs to represent relationships between languages. We will analyse different scenarios that have been built to understand what can be found and how it compares to our current knowledge of this situation. These scenarios will consist of combinations of different groups of languages or language families.

3.2 Methods & Models

3.2.1 Data

In this section, we will discuss how we determined which data set to use. As previously explained, this step is crucial because all the results will be based on how accurate and representative the data is for the language. Taking a database that contains concepts that are loan words in many languages is a bad idea for example. Gathering the right data the first step in the methodology, but it is important that it needs to be usable for the next step, where we will need to form some summary statistics based on this data. We also have to remember that we want language data that correctly represents a linguistic variety or language. This also means that the size of the

samples from different languages have to be similar. As a side note, there is no variation in the data between speakers, unlike genetic data, where there exists some variation between members of a species. This means that it is enough to have only one sample (one speaker) from each language, compared to genetic data.

Types of databases

There are two main categories of linguistic data one can focus on when searching for linguistic data that can be used in our study: we will present IELex and LexiRumah as examples of these two categories. The first one (which is the one we will use in this research) is data that only contains languages from the same family and the second one is data that focuses on many different languages within a region, without looking at the which family they belong to.

As we have previously seen, the coalescent assumes that there is a most recent common ancestor to all the individuals simulated. When the data only contains languages from the same family, this assumption is satisfied. By definition we know that is not needed here because we know the family has a common ancestor. The IELex database is a good example of this type of data: it contains only languages that are inside the Indo-European language family. The common ancestor in this case being Proto-Indo-European (Hammarström et al., 2021). The languages in this data are not geographically close to one another, meaning that there is not as much direct contact between all the languages as in the previous example (linguistic exchange between Irish and Iranian languages would be limited by the distance between two speakers of these languages). Nevertheless, we can still look at different language groups inside the Indo-European language family to research how the languages inside and between two or more language groups influence each other (and this time, without having to make an assumption common ancestor).

An example of the second type of data is LexiRumah. LexiRumah is an online database (Kaiping & Klamer, 2018) that is a lexicon of languages, where the area of focus is the Lesser Sunda Islands in eastern Indonesia. The lexicon contains 357 linguistic varieties belonging to 11 language families, mainly consisting of the Timor-Alor-Pantar language family and the Austronesian family. This type of data is an interesting study case for borrowings between languages, because it allows us to look at how languages from different language families exercise influence one each other. To use the coalescent with this type of data would make a big assumption, because it would mean there is a common ancestor at the top of the different families. This is inherent to the coalescent technique, as it coalesces until there is only one most recent common ancestor.

IELex

IELex (`http://ielex.mpi.nl/`) is a database containing concepts from the Swadesh list (Swadesh, 1955) for Indo-European languages (Dunn, 2012). The Swadesh list is a list of (originally) 215 basic concepts that can be used to compare languages. A letter indicates cognate classes inside the IELex database. Unfortunately, the website hosting the database is down, but other websites host copies of the file. The file is a `.csv` containing 5 columns: Language, Meaning, Phonological Form, Cognate Class and 'cc'. Language is the column that contains the language, 'meaning' represents a concept written in English. The phonological form is the International Phonetic Alphabet (IPA) form of the concept in a certain language. Cognate class and "cc" are very similar, cognate class is a letter that can be used to find out which concepts are cognates and 'cc' is the value from the column "meaning" plus the letter from cognate class, separated by a ':'. The IELex database that we use in this research contains 52 living and extinct languages and 208 different concepts. Any language can have multiple forms for one word, but we only take the first synonym from the database for simplicity. A sample of the database is given in table 3.1.

Language	Meaning	Phonological Form	Cognate Class	cc
Ancient Greek	sharp	oksýs	C	sharp:C
Greek	sharp	ˌkǫftẹˈrǫs	J	sharp:Jine
Classical Armenian	sharp	sowr	D	sharp:D
Armenian Eastern	sharp	suɾ	D	sharp:D
Ossetic	sharp	tsɨrʁ	A	sharpA

Table 3.1: Sample of IELex database. Form and cognate class of the word ”sharp” in five different languages.

IELex is interesting because it does not contain languages from other families, even when the languages are geographically close to each other. For example, Hungarian is a language that is spoken in Hungary, but it is not available in the data. The fact that IELex is strict has the advantage that no common ancestor assumption has to be made. Together with the interesting complete cognate data for the concepts, this makes it suited for an analysis using the coalescent theory.

3.2.2 Libraries

In this section, an overview of all the libraries specifically needed for the research is provided.

Msprime

`Msprime` is a library made to efficiently simulate coalescent models in Python in a simple way. It inherits a lot of aspects from the original `ms` program, which is a Monte Carlo program to generate samples from a population following the Wright-Fisher neutral model (Hudson, 2002). According to the authors of `msprime`, it performs significantly faster than most Sequentially Markov Coalescent approximations, without having to rely on approximations (Kelleher et al., 2016). `Msprime` relies on a data structure called a tree sequence that is implemented by `tskit`. The ease of use and the fact that it can be integrated into a Python program that uses classic Python libraries is the reason we chose for this library.

We use version 0.7.4, because this was the latest version of `msprime` at the start of the research. During the research a new version of the program has released (version 1.0.0) with major changes to the core of the library. One of the biggest problems with the 0.7.4 version of the library was the fact that `msprime` used the infinite sites model. This means that in `msprime`, each locus of the genome undergoes at most one mutation. (Kelleher et al., 2016) This has an important consequence, because if one locus can only undergo one mutation, it cannot change back to its original allele, or even change to a totally new third allele. For a genome, this is not a real problem, but for our representation, where one allele represents the cognate class of a concept in a language, it is. The new version of `msprime` introduced a finite sites model.

For every version of `msprime` before version 1.0.0, a conversion has to be made if a finite sites model is required. We had to to this conversion because we use version 0.7.4. To convert an infinite sites model to a finite sites model, a conversion of every mutation has to be done. In `tskit`, mutations are shown as decimals that are between two integers. To do the conversion from an infinite sites model to a finite sites model, you have to go over every mutation site in the genome (a decimal number, because of the infinite sites model) and bind that mutation to a finite site (an integer). Once this is done every site in the genome has to be given a random ancestral state from a pool of possible states, chosen by the user (for example 0 and 1 or A, C, G and T). Now the parents of the mutation have to be computed (this can be done with

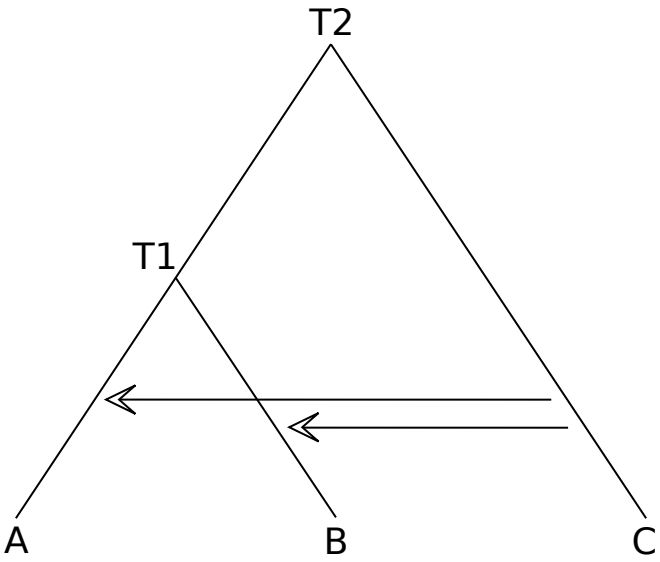


Figure 3.1: Tree shape of language tree where A, B and C represent the languages, T1 and T2 represent the population split times and the two arrows represent borrowing.

`compute_mutation_parents` from `tskit`) and derived states have to be computed. The chosen derived state is chosen randomly from the given states, and it cannot be the same as the previous state. Once this is the done, the tree has been adapted to a finite sites model.

3.2.3 Language Representation Using Coalescent Theory

Now that we understand what library is used and how it should be adapted to our needs, we need to get a better idea of how we should represent the problem we want to solve.

Language Tree

Following the example of Thouzeau et al. (2017), we use three different languages to form a tree. This is the optimal number of languages, because these triplets can be reused. Choosing two languages would give no information and using four languages would not be the minimal number of languages. It is also more feasible to only use three languages, because computing and optimizing a whole tree in one process would consist of many input parameters which all have to be optimized. This is too computationally expensive to be a feasible solution.

The shape of this tree is constant: two languages or linguistic varieties are grouped together under a node representing their most recent common ancestor and the parent node of these two languages are combined with another language into another node. This shape can be seen on figure 3.1. The language tree is the main structure that will be used many times to visualize the result of the process. As can be seen in the figure 3.1, there are five symbols and two arrows that give information about the tree: these are the inputs for a simulation. The letters underneath the tree represent the three languages we are studying. ”T1” and ”T2” represent population splits times and the two arrows represent borrowing. It is very important to remember that ”T1” must be smaller than ”T2” in all cases. For example if ”T1” is equal to 50, this means

that the most recent population split between language A and B happened 50 generations ago. Because this is the most recent population split, it must be lower than "T2".

It is important to mention that for combination of three languages we choose, there exist three combinations, because inverting A and B in figure 3.1 will not result in any meaningful changes. The three combinations for languages A, B and C are: [[A, B], C], [[A, C], B] and [[B, C], A].

As previously mentioned, `msprime` uses `tskit` to represent trees that are the result of the coalescent simulation. Every spot in the genome has an underlying tree to represent mutations and ancestry. If we keep the same tree shape for every site in our genome, we can better analyze the impact of changing parameters of the simulation. For `Msprime` specifically, population splits and borrowing are represented by demographic events. The population splits are given as a number that indicates the time at which the event occurs (in amount of generations ago). Borrowing events happen after the most recent population split.

Genome Sequences for Languages

Conceptually, what we will have to do at this step of the research is to find a way to interpret the results from the coalescent simulation as linguistic data (more specifically cognates). We have previously been able to give parameters to our model and it has returned a most recent common ancestor and some mutations along the tree branches. We have to access this mutation data in order to find out what the final cognates of the simulated data look like. Some cognates will have never mutated, while others will be different from their original form. Because we work with the coalescent, we assume that at the most recent common ancestor, all cognates were equal. In our model, the three has three languages, which means that we are able to construct three cognate data lists, one for each language.

In our implementation, we have to do some work to be able to use this idea. As explained in a previous section, `msprime` (in version 0.7.4) by default works with a infinite sites model to simulate genetic data from a coalescent simulation. When we adapted this to support a finite sites model, but we did not solve the problem where every site in our genome follows the same ancestry, nor did we solve the problem of the amount of genetic loci wanted. To solve the first problem we have to create population splits that involve three populations. As shown on figure 3.1, there are two splits required per language tree, one from an ancient population that split at time "T2" into population C and a combination of population A and B. Later on, at time "T1", population A and B split from each other to form two separate groups. How do we model this in `msprime`? The new version has an explicit demographic event called "`add_population_split()`", but in the older versions every demographic event has to happen with the "`MassMigration`" event. This event has a parameter called "`proportion`" which is the chance that a lineage from the source population migrates to a destination population. If this proportion is set to 1.0, the event is considered a population split. The code for the population splits can be seen in snippet 3.1

Listing 3.1: Inserting population split T1 and T2

```
demographic_events = [
    ...
    msprime.MassMigration(time=t1, source=0, destination=1,
        proportion=1.0),
    msprime.MassMigration(time=t2, source=1, destination=2,
        proportion=1.0)
]
```

Now that every part of the genome follows the same ancestry, we have to set the amount of sites in our genome to the number of concepts we want to analyse. This task can be solved by simply defining our own `RecombinationMap.uniform_map` with the length equal to the number of loci in the genome. The rate shown in 3.2 is the rate of recombination per unit of sequence length along the chromosome. The rate of recombination is equal to one because it makes it so "breakpoints between trees now occur exactly at the integer boundaries between these loci" (Kelleher et al., 2016).

Listing 3.2: Recombination map for simulation

```
recomb_map = msprime.RecombinationMap.uniform_map(length=nrconcepts,
    rate=1, num_loci=nr\_concepts)
```

All of this work results in a number of sites equal to the number of breakpoints given as `nr_concepts`, with each site (where at least one mutation happened) having a symbol which expresses in our case to which cognate class a concept in a certain language belongs to. Only sites where a mutation happened need to be associated with a symbol, because in the other case all three concepts are cognates, which means that they all have the same symbol. To give an example of what it would look like if we try to simulate 5 concepts and print the results, as shown in snippet 3.3

Listing 3.3: Cognate data as a result of a simulation

```
B–AD–
B–AB–
A–AC–
```

These results show us that two sites have no mutation: site 1 and 4 (if we start counting from 0). These are marked with a special neutral symbol '-'. The other sites have undergone mutation and the end result shows some similarities and differences. It is possible for a site to mutate twice, which can have as a result that all three concepts are cognates (even with mutation). This happens on site 2 in the example given. This phenomenon happens rarely, because there are many possible cognates a certain cognate can mutate to.

This is the simulated version of going over every shared concept of three different languages and writing down three rows representing the cognate classes of these shared concepts. This is shown as in the snippet below:

Listing 3.4: Cognate data as a result of a simulation

```
Concept: sharp
Language: Polish. Cognate class: C
Language: English. Cognate class: B
Language: French. Cognate class: I
Concept: water
Language: Polish. Cognate class: B
Language: English. Cognate class: B
Language: French. Cognate class: E
Concept: snow
Language: Polish. Cognate class: B
Language: English. Cognate class: B
Language: French. Cognate class: B
Result:
```

```
Polish:  CBB
English: BBB
French:  IEB
```

One observation that can be made is that the concepts have the same source in the tree, so we could say that because there is always only one source, the results are always cognates. This is not true, because having the same source does not mean that every word in a language has the same etymology.

3.2.4 Coalescent Parameters

To start simulating ancestries and mutations with `msprime`, we have to define some parameters first. In the next sections, we will be covering different parameters that have to be configured in order to start simulating cognates with `msprime`.

Population Configuration & Migration Rate

Population configuration is a parameter that introduces new populations in the coalescent simulation. In our case, we want to introduce one population for every language (three in total). Each population is of size 1, because each population represents one word.

The migration rate is closely related to the population. Because our population size is only of size one, we do not want to add any migration to the coalescent simulation. However, this does not mean that we do not want to add any `MassMigration` events, because they represent a probability that a lineage migrates to a destination group of the population (not a part of the population). We also do not add any growth in the populations. Finally, at the start, we sample the whole population by taking a sample size of one.

Mutation

Mutation is an impactful parameter for the coalescent simulation. It is difficult to find an objective mutation value that is general enough for all languages in the our data. Some languages evolve faster than more remote languages, and we want to keep the mutation the same for all languages, because the results would be inconsistent otherwise. The task will be to find a good value for mutation rate that can be used in every possible situation. If it is too high, no valuable information can be retrieved from the result of the simulation. If it is too low, there is no change that can be measured on the genome, which will result in very little information. A value between 0.001 and 0.01 should be good enough to get the most important information out of the simulations. This value was chosen so that most concepts had at least one mutation in their ancestry. It is not too important to get it perfectly right, because the only parameter that will be affected is the ”T2” and ”T1” from our language tree. The lower the mutation, the higher these two population splits will need to be, so that we get enough mutations. Too high of a mutation rate will make these population splits almost at the same time. The details about this will be explained later on.

Population Splits

Population splits can be seen in the figure 3.1 annotated by ”T2” and ”T1”. These two values will be very important results later on for many different reasons. Population splits are events that model a population splitting in two newer populations. All previous mutations that happened on the ancestral population can be found in the newer populations. In our case, we will use

population splits to simulate two language being different enough to be classified as two separate languages.

In `Msprime`, population splits are defined by `MassMigration` events with a proportion of 1.0, meaning it has a 100% chance of moving to a new population. Population splits are defined from bottom to top, so the sources and destination seem to be inverted as can be seen in the snippet 3.1. The newer version of `msprime` allows to define an event that is a specific population split, which helps visually separate the different events in the code.

Admixture

Admixture events will be one of the most important ways of simulating borrowings or the influence of one language to another. This is very important to understand, because we will sometimes use admixture events as if they are the same as borrowings. In genetics, admixture is means that two previously split lineages mix again (Yang & Fu, 2018). We will use this definition for languages, where we will want to express languages "mixing" some proportion of their concepts.

In `Msprime`, admixture events can be defined by a simple `MassMigration` event, but with a proportion lower than 1.0 (and higher or equal to 0).

As can be seen on figure 3.1, only two arrows are displayed for influence. Why not six, one from every branch to another? The two arrows from A to B and B to A can easily be removed by lowering (or increasing the value of) the population split "T1". This will intuitively make both A and B more (or less similar), because they will have less time to have different mutations.

Two other arrows from A to C and B to C can also be removed, because, similarly to the previous case, they can be replaced by lowering or increasing the "T2" population split. This is a bit more difficult to understand intuitively, but without going into much detail in this section, if we want A or B to be more similar to C, we should make the population split "T2" happen sooner.

Why can we not remove the last two arrows? Because these arrows specifically target one of the two populations under population split "T1". This is a unique property that cannot be replicated by changing the times at which population splits happen.

3.2.5 Summary Statistics

In this section, we will explain what summary statistics are computed to compare results from the coalescent simulation and the real data. Given some input parameters to a tree, we can simulate the tree and look at the result that this execution gives us for a specific set of parameters. The data from an execution of a coalescent simulation are strings of characters. We want to know evaluate this simulation by some objective function. We could compare each character with the real data, but this will never result in a perfect match.

For this reason, we have to look for statistics that summarize the aggregate data in a representative way. Thus, the summary statistics should be easily computed and be representative of the data. The goal of these statistics is to have an objective function that can be used to compare how far or close we are to the real-world data. This step will help us transition from the step where we built the language trees to the next step, which will be about getting information out of the language trees. Most importantly, we need to be able to get the relevant information out of the result of a simulation in order to give a verdict on how accurate the simulation was when compared to the real-world. In other words, We have to be able to concisely decide if a simulation represents the real-world data correctly.

Metrics

As in the study made by Thouzeau et al. (2017), the summary statistics computed should be able to represent the proximity between the observed and the simulated data, so that we can select the most likely scenario from the parameters given to the simulation. There are many different possible statistics that can be interesting, such as the mean number of cognate classes per concept, or the variance of the number of cognate classes per concept, and many more. For this research, we will stick to three summary statistics. We chose three statistics because it gave us a good result, which will be shown later. More summary statistics can be chosen, but they should be present in the results of the simulations and the real-world data. For example, the average number of mutations cannot be seen in the real-world data, but could be computed from the simulation.

The first summary statistic is the number of pairwise differences between cognate classes for each concept of two languages, divided by the total number of concepts. This is of course done three times, because we have three pairs of languages. The amount of times this needs to be computed depends on the amount of languages there are. The formula is: $\frac{\mathbf{L*(L-1)*(L-2)}}{\mathbf{2}}$, with L as the number of languages.

The second summary statistic, although not as useful for the analysis, is a useful metric to get an idea of how many cognates need to be simulated and how high or low the mutation rate should be. This statistic is the mean number of cognate classes per concept. The highest value possible could be equal to the number of linguistic varieties there are (three in this case). This was the case for the word "sharp" in Polish, English and French, as shown in code snippet 3.4, all three words have a different linguistic derivation. The lowest possible value is 1, every concept is a cognate. This was the case for the word "snow" in Polish, English and French, as shown in code snippet 3.4. The third statistic is the variance, which is closely related to this one. This is the squared deviation from the mean, which in this case means the deviation from the mean number of cognate classes in total. A high number would mean that there are a lot of concepts with a high and a low number of cognate classes. A small variance means that most concepts have the same amount of cognate classes (which will be equal to the mean).

Example of Computing Summary Statistics

The easiest way to explain this is to show how to compute this statistic for three example linguistic varieties or languages. This process can be applied on to real-world data or simulated data. The table 3.2 shows a typical result from the analysis, only here for five concepts instead of hundreds of concepts. The first statistic exists out of three parts:

$$C_{L1,L2}= \frac{\mathbf{0+0+1+0+0}}{\mathbf{5}} = \mathbf{0.20}$$

$$C_{L2,L3}= \frac{\mathbf{0+1+0+1+1}}{\mathbf{5}} = \mathbf{0.60}$$

$$C_{L1,L3}= \frac{\mathbf{0+1+1+1+1}}{\mathbf{5}} = \mathbf{0.80}$$

The way to compute these numbers is not complex: count all the differences between concepts and divide it by the amount of concepts. This statistic gives us a good way to see which languages are more similar to each other. A low difference means that the linguistic varieties are similar to each other. This value can range from 0.0 to 1.0.

The second statistic is computed by counting the number of cognate classes for each concept, and dividing this by the number of concepts. For our example in table 3.2: $\frac{\mathbf{1+2+2+2+2}}{\mathbf{5}} = \mathbf{1.80}$.

	C1	C2	C3	C4	C5
L1	A	B	D	F	H
L2	A	B	E	F	H
L3	A	C	E	G	I

Table 3.2: Table of Concepts (C), together with their linguistic variety (L) and resulting in cognate class A to I

The third and last statistic is computed with this one. The variance can be computed like this: $(\mathbf{1.80-1})^2+(\mathbf{1.80-2})^2+(\mathbf{1.80-2})^2+(\mathbf{1.80-2})^2+(\mathbf{1.80-2})^2=\mathbf{0.8}$

To get something meaningful as a result of the coalescent simulations, we have to compare the summary statistics from the simulation to the summary statistics from the real data. This comparison can simply be done by computing the Euclidean distance between the statistics. Of course, you have to remember to standardize the data if you combine statistics that are not on the same scale. This has to happen sometimes because some statistics like mean number of cognate classes would for example, weigh more in the formula for distance. To solve this imbalance in those cases, we can divide the number by their range.

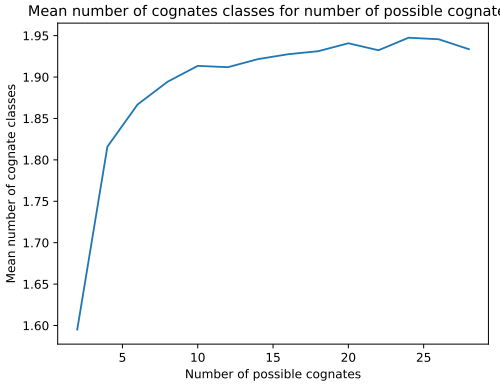
We also have to remember that not every statistic is important at every stage. For example the mean and variance metrics will mostly be determined once in the research. We will try to determine how many possible cognate classes a value will be able to take. This is shown on figure 3.2. The values on the graphs show that around 15 to 25 possible values for the cognate classes, we reach a stable value for the summary statistics for mean and variance. This value basically means that we with that many possible values, we avoid having too many cases where a cognate mutates back to a form it has already taken, because this does not happen often in real-life. In other words, this helps us ensure that we have enough cognate class possibilities. We computed this for two different cases, one where the population split times were close to each other and one where the population split times were far from each other. In both cases, the results are similar, which means that we will use a value of 25 in our research for the amount of possible cognate classes. The amount of cognate classes in the IELex database is equal to 122, but the figure 3.2 shows us that we have already reached a stable value after 25. A stable value means that if we add more possible cognates, it will still give the same value for these summary statistics.

Finally, each execution of a simulation will yield results that can differ from time to time. For this reason, to get accurate summary statistics, we have to average results over multiple executions. But how many executions give an accurate result? Figure 3.3 shows us that around 200 to 300 executions we get a stable result around between 0.340 and 0.345. A stable result is needed to get repeatable results, so running enough executions is import to get the most precise results.

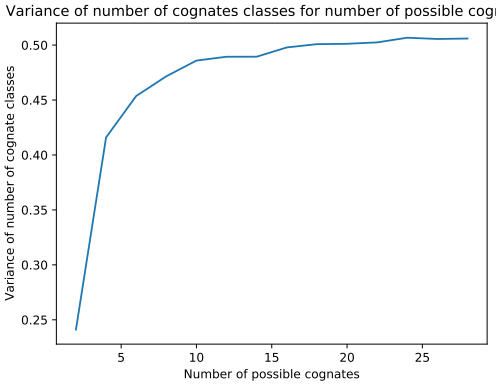
3.3 Minimizing Differences in Summary Statistics

Our model has four inputs: time of two splits and the proportion of two admixture events. Our outputs are summary statistics that can be computed on real and simulated data. The goal now is to optimize the four inputs, which means that we want to minimize the difference between our simulated data and real data summary statistics. The four inputs are population split time T1 and T2, together with the borrowing from language C to A and from language C to B.

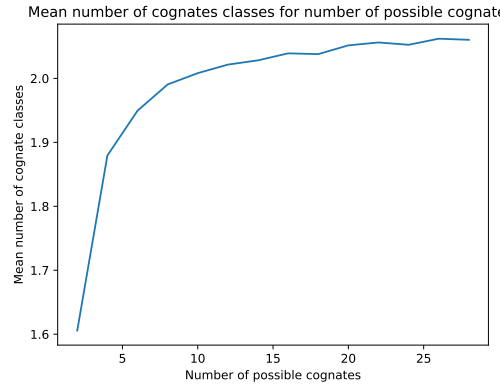
In the first part of this section, we will try to find good input values for starting the optimization process. We will start by only looking for the best T1 and T2 values, because they are the easiest to guess right and they provide us with a good start. In the second part we will optimize



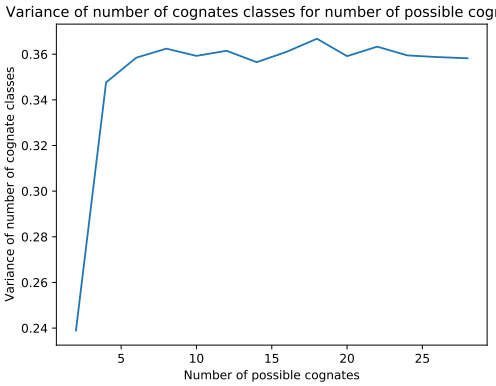
(a) T1 = 50 | T2 = 100



(b) T1 = 50 | T2 = 100



(c) T1 = 30 | T2 = 180



(d) T1 = 30 | T2 = 180

Figure 3.2: Plot that shows the change in mean and variance when adding more possible cognates in two example cases. A stable value is important for because it means that there are enough values in the pool of possible values to represent the concepts in the simulation. We reach a stable value between 15 to 25 in all cases, because it does not move that much at that point.

all of our inputs based on these starting values. The optimization process is difficult, because there are specific bounds that we cannot violate, which is why we will introduce a optimization method specific for our case.

For this method, we want to minimize the differences in order to find the most accurate input parameters for a model. The difference in output of the model and the real-world data should be minimized to have the most accurate model. The outputs are represented by the summary statistics and the inputs are the only things that can be changed in order to change the output.

With the accurate inputs we will be able to construct graphs representing the relations between the languages. These relations cannot be known by looking at the real-world data, but they can be computed with the help of the inputs of the coalescent as will be seen in the next step.

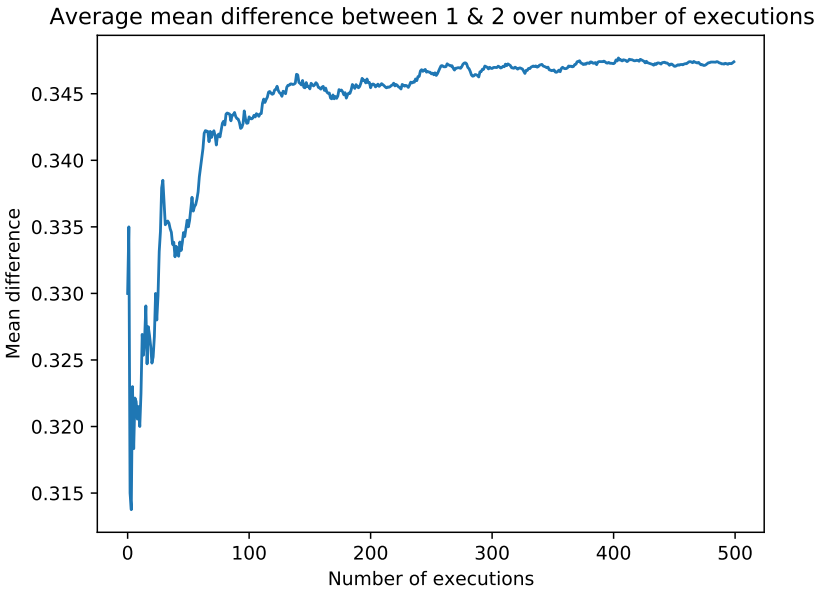


Figure 3.3: Average value of the mean difference between the cognate classes of the language 1 and 2 after a certain number of executions. At the start, the mean of this summary statistic fluctuates a lot, but after 200 simulations, the value stabilises around 0.345. The same plot could be shown for the other two summary statistics, but this shows enough to understand how many simulations have to be run in order to get a stable result.

3.3.1 Computing Preliminary Population Splits

In the previous sections, we have mentioned several times that there are two population splits "T1" and "T2". We also know how to calculate the distance between the mean difference between the cognate classes of the languages (the three numbers). What we need to compute in this step is some value for the two splits that is close enough to the final value. In the next step, we will add admixture events, which can make it so we have to change population splits again. Having a rough estimate will help us in this next step to make a better calculated guess to know which parameters have to be changed.

The easiest way to make a guess without having any information except for summary statistics, is to try the values out and see what the results are. In the following example, we will try out the values for "T1" and "T2" which range from 1 to 200 (in increments of 10). This range is chosen after running some experiments with a mutation rate value of 0.005. This does not mean that the maximum value for "T2" will be 200, but that the initial guess will maximally be that high. We average out ten simulations, even when previously shown that it is important to run at least 200 simulations, because this value is only used as a preliminary value and will be more accurate in the next step. The example for Dutch (as language A), English (as language B) and French (as language C), can be seen in figure 3.4. As a reminder, we use the differences between mean difference of languages (the scores that range from 0.0 to 1.0) as a way to score the result, so that means that low scores are close to the real data. In figure 3.4, we can definitely see a cluster, with the absolute minimum around T1=41 and T2=121. The bottom left are the values that are impossible, because "T1" would be higher than "T2", which would create an impossible language tree. The way to interpret these results is as follows: there was a split between English and Dutch 41 generations ago and there was a second split between their most recent common ancestor and French 121 generations ago. Generations here are not absolute values, as in, we cannot compute it with real years. The ratio between the two values is much more important, because they are not affected by mutation rate. In other words, if mutation rate would increase, the time values would also increase, but the ratio between them will stay the same. The starting language tree can be updated with those values, as shown on figure 3.5. The values could potentially be related to real-world data if there was a known split date between Dutch and English, but this will not be explored in this research.

3.3.2 Computing Admixture & Final Splits

Last section we discussed how to find the preliminary results for population splits, so that the summary statistics of our model approached the summary statistics of the real-world data. This was preliminary because we tried finding the best score (lowest score) by simply computing what the score was for simulations where we only changed T1 and T2. The best score in the previous step step does not mean that we have achieved the most accurate input values possible. What we have achieved in that step is a good starting value for this following step. We want to include the borrowing to create changes in the summary statistics that were impossible to do previously. In other words, we previously only used the four first columns of table 3.3. This was important to compute so we can have a good starting point for the next part.

The goal of this part will be to get two new values for the admixture events from language C to language A or B, as shown on figure 3.1. Together with this, we will try to perfect the values of the population splits that we computed in the previous section, but this time, we will take the admixture events into account. As previously explained, admixture events are used to simulate the exchange from one language to another.

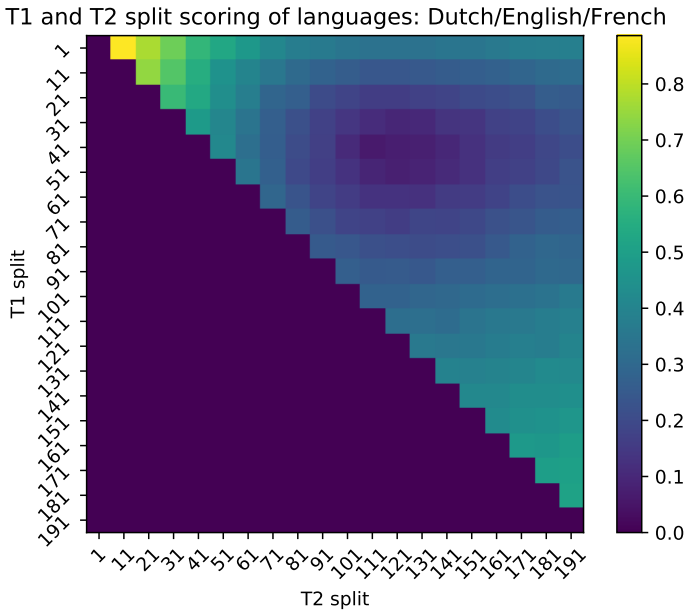


Figure 3.4: Heatmap showing scores (Euclidean distance between all differences) for split times. Every square represents the average score over 10 runs for a combination of split time T1 and split time T2 for the language tree. A low (i.e. darker) score shows a smaller difference between the summary statistics of the real-world data and the simulated data. A low score is thus considered better. The bottom left side is not possible, because T1 would be greater than T2.

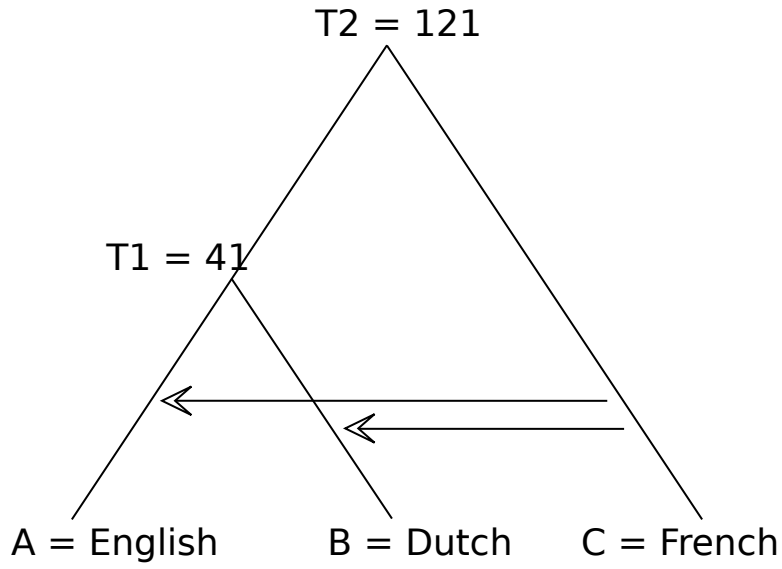


Figure 3.5: Language tree with split values filled in from the preliminary population split times computation (heatmap shown on figure 3.4).

	↑T1	↓T1	↑T2	↓T2	↑C →A	↓C →A	↑C →B	↓C →B
A/B	↗	↙	-	-	↗	↙	↗	↙
B/C	-	-	↗	↙	-	-	↙	↗
A/C	-	-	↗	↙	↙	↗	-	-

Table 3.3: Table that shows how changing one of the inputs will impact the summary statistics. Arrows show an increase or decrease in the value of a specific summary statistic shown on the left. '-' means no change in the value. In our experiments, we want to use these increases and decreases to get closer to the real-world data summary statistics.

Problem of Classical Optimization

We have to remember that this is not a simple optimization problem. We have a noisy function, which is the distance between the simulation summary statistics and the real-world statistics. This could in theory be solved by a simple gradient descent that looks for the minimum (distance between the summary statistics). The problem is that we have bounds which are also dependant on the inputs. There exists a bound that dictates that T1 cannot be larger than T2 and the inverse bound exists too. These bounds are not constant, which makes the problem more difficult to solve. Fortunately, we do have prior knowledge about how the model reacts to certain changes in parameters that we will use to solve this problem in a quicker way. This will be the topic of this section. For these two reasons (difficult bounds and speed), we have decided to use our own optimization method. Instead, we will solve smaller optimization problems that are bound by constant values together with our knowledge of the impact of the inputs in order to complete this problem in a quicker way.

Impact of Input Parameters

Before looking at the two ways to approach to this problem, we have to understand what impact these four inputs (T1, T2, C to A and C to B) have on the summary statistics. The table 3.3 shows what the impact is.

These results can be thought of logically. Increasing "T1", will put the most recent common ancestor for language A and language B further back in time, which will increase the difference between A and C (the statistic representing that). In a similar manner, increasing "T2" will cause the most recent common ancestor to be placed even further back, causing a bigger difference in the language from both major branches. The two admixture events will decrease the difference between the language C and the destination language, but this will cause an increase in the difference between the languages in the tree for A and B.

How do we get closer to the real data summary statistics with this information? Every triplet of three languages returns three numbers indicating how close each language is from another. An example can be thought of for three languages: English, Dutch and French. We denote the mean difference in cognates (calculated in the same way as in section 3.2.5) for concepts as $\mathbf{C_{L1,L2}}$ for language L1 and language L2. For our example, this results in $\mathbf{C_{English,Dutch} = 0.356}$, $\mathbf{C_{English,French} = 0.665}$ and $\mathbf{C_{Dutch,French} = 0.728}$ (all rounded up to the third decimal). Every time we run 200 or more simulations (one simulation generates cognates for all concepts) and average them out, we get a result with these three important values. We will use the difference between the simulated values and the real-world values to get a measure of how far we are from the desired accuracy. This could be 0.01 for example, if we want to the maximum difference between the summary statistics of the simulated data and real-world data to maximally be 0.01. If we look back at table 3.3, we can see how we can influence the simulated values by manipulating the

inputs. In our example, if our simulated data results in $\mathbf{C_{SimEnglish,SimFrench} = 0.700}$, we should lower it, for example by lowering "T2" or increasing the proportion of the admixture event from French to English.

This means we have to solve two problems. The first problem is about what we have to change in the input to get the wanted results. What should we do if we need to increase $\mathbf{C_{SimEnglish,SimFrench}}$, decrease $\mathbf{C_{SimDutch,SimFrench}}$, while causing no change to the value of $\mathbf{C_{SimEnglish,SimDutch}}$, as shown in the following output?

Listing 3.5: Difference between summary statistics of the real data and simulated data

```
Real: English <-> French = 0.665
Sim.: English <-> French = 0.620    - Action needed: Increase
Real: Dutch <-> French = 0.728
Sim.: Dutch <-> French = 0.780      - Action needed: Decrease
Real: English <-> Dutch = 0.356
Sim.: English <-> Dutch = 0.350     - Action needed: None
(Precision: 0.01)
```

The second problem we have to solve is in what capacity does a small change to one of the input values impact the output values. For example, if I know I need to increase T1, I still need to know by how much. In the example snippet 3.5 shown above, we look for an exact number that represents how much we have to add to one of the inputs to increase the difference between English and French.

Solving the First Problem The first problem can be solved in two ways. A first approach could be to use a solve a set of equations and inequalities with some constraints. To understand this, we have to look at the table 3.3. If we look at this table, we can construct equations and inequalities with the four inputs, that result in the output that we want. We denote the difference between the summary statistic of the real data and the simulated data with $\Delta C_{L1,L2}$. The equations would look as follows, where A_{T1}, A_{T1}, $A_{(C\to A)}$ and $A_{(C\to B)}$ represent what a change of one unit (0.01 for admixture rate and 1 extra generation for population splits) does to a certain summary statistic. As an example value, we could take an addition of 0.01 proportion for the admixture rate from C to A, and we could see an increase to the summary statistic representing the difference between language A and B of 0.005 and decrease the difference between A and C of 0.005, following the changes shown in table 3.3 (the values are examples, they will be defined later). How to find these values is the topic of the second problem.

- $\Delta C_{L1,L2} = x_1 * A_{T1} + x_3 * A_{(C\to A)} + x_4 * A_{(C\to B)}$
- $\Delta C_{L2,L3} = x_2 * A_{T2} - x_4 * A_{(C\to B)}$
- $\Delta C_{L1,L3} = x_2 * A_{T2} - x_3 * A_{(C\to A)}$
- $(x_1 * A_{T1}) + T1 < (x_2 * A_{T2}) + T2$
- $0.0 \leq x_3 + (C \to A) < 1.0$
- $0.0 \leq x_4 + (C \to B) < 1.0$

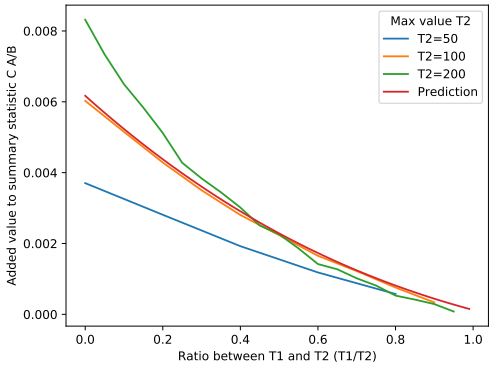
The three first equations can also be relaxed by allowing the change to be a bit more or less than the value. This can be done for example by introducing two inequalities and add/subtract to the delta a value that represents the wanted precision. The last three inequalities are constraints that must be satisfied by the first three.

An alternative approach, which is the one that we have chosen, would be, instead of solving the equations, to minimize the Euclidean distance between the 3D (in this case because we use three languages) point representing the summary statistics of the real-world data and the simulation. The problem with the previous approach is that sometimes, the perfect solution did not exist, which was then solved by replacing the equations with a set of inequalities. Here we are going to try to get the two 3D points as close as possible, even if the two points do not overlap (which would mean that we found a perfect solution). This method can easily be done with SciPy's minimize function (Virtanen et al., 2020). We used the modified Powell algorithm (Powell, 1978) in our study, but any other method that supports constraints can be used.

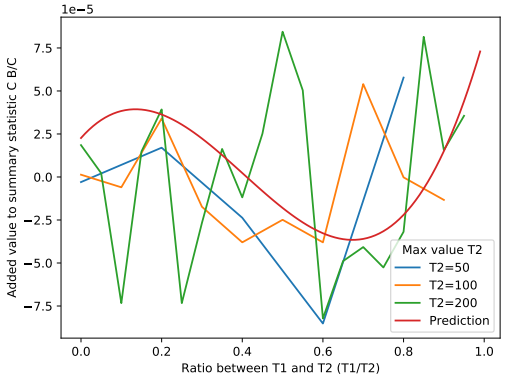
Solving the Second Problem Now, we have to solve the second problem. The problem of finding what A_{T1}, A_{T1}, $A_{(C\to A)}$ and $A_{(C\to B)}$ represent is not easy. The biggest problem is that the values depend on each other, so that it makes it difficult to take one value as the value that can be used in every situation. A better way to solve this is by experimenting with different values and trying to fit a polynomial or linear regression. This regression should have as input the ratio between "T1" and "T2" ($\frac{T1}{T2}$) and should return a value that represents what change adding one extra unit gives us. For example, if we have $T1 = 50$ and $T2 = 100$ then our input would be $\frac{1}{2}$ and our input would be some value α. This α will be equal to some unit that can be put in the equations shown above (the values A_{T1}, A_{T1}, $A_{(C\to A)}$ and $A_{(C\to B)}$). The figures 3.6 and 3.7 show what adding 0.01 proportion to the admixture event from language C to language A or language C to language B does to the summary statistics. We can compare the table 3.3 to the results shown in the figures and see that the results correspond to the predictions made. There are four lines in every figure: the red line is the prediction made with a fitted line, using the real values gotten with $T2 = 100$, the two other lines are real values from simulations with a different T2 value. In this case, we trained the regression line with the values of $T2 = 100$. This value means that we simulate multiple examples with "T2" fixed and move "T1" closer and closer to "T2". The values received by this are shown as an orange line. The other two lines are two more examples, where "T2" was fixed at 50 and 200, to analyze what impact this has on the results. This regression is a third degree polynomial regression for all the increases in proportion of the admixture events. Because we are not going to use this regression for values outside the bounds [0, 1], we can almost find a perfect fit. We look for the best fit by estimating the errors using the method explained in Richter (1995). This method uses the square root of the covariance matrix to find error bounds. We find almost no difference in errors between the different polynomials for the data displayed on figures 3.6a, 3.6c, 3.7a, **??**. The biggest differences can be found in the data found on figure 3.6b and 3.7c. There, a third polynomial fits best.

It becomes very clear by looking at figure 3.6a and 3.6c that the ratio between "T1" and "T2" does not explain enough, because there is a clear difference. The blue line in this figure follows a line, while the green line follows a curve. This is especially true when "T1" is much smaller than "T2". The easiest way to deal with these inaccuracies, is to find some scaling factor σ that decreases the output value from the regression when "T2" is smaller than 100. An example of a scaling factor that we use is $\sigma = (T2-100)*0.0001$. The figures for the increase of the proportion of 0.01 in the admixture event C to B show the same type of inaccuracies.

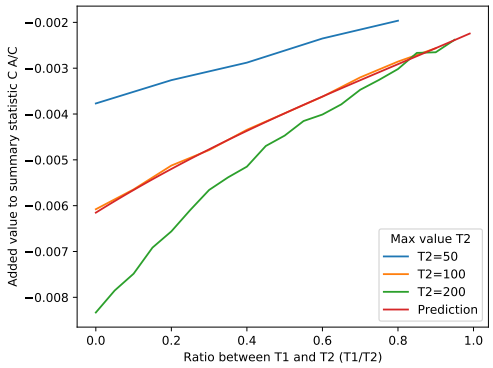
For T1 and T2, the same graphs can be shown for the increase (or decrease) of the population split time. The figures 3.8 show three linear regressions that fit the simulated data. The data represents for figure 3.8a $\frac{T1}{T2}$ where "T1" is slowly increased until it reaches "T2. The data on figure 3.8c and 3.8b occurs when we keep "T1" fixed instead at a value of 20 and slowly increase "T2" from 20 to 200. A linear regression was chosen here because the polynomial regression formed curves, overfitting the data.



(a) Change C to A



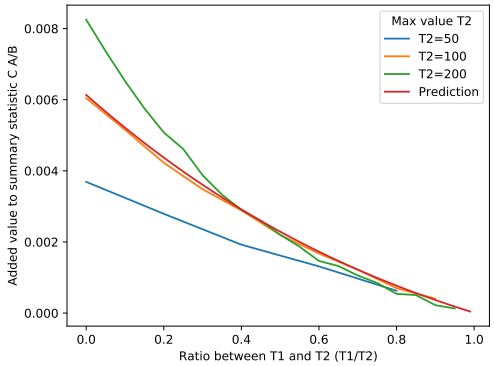
(b) Change C to A



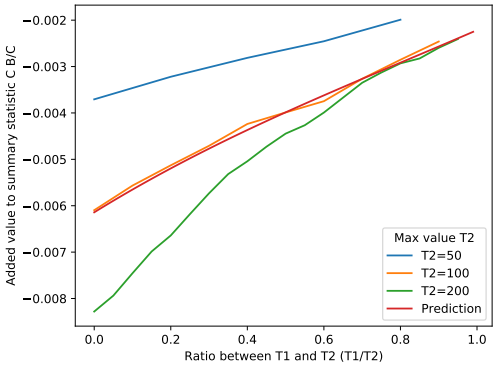
(c) Change C to A

Figure 3.6: Solving the Second Problem: Increment for summary statistics, given an increase of proportion of 0.01 in the admixture event C to A. Average results for 200 runs. We see the green line show that changes in C to A have a higher impact on the summary statistics that show the difference between A and B and A and C. The blue line shows less impact. The impacts are greater when the ratio between T1 and T2 is lower. The fitted line shows a good middle ground, but at a lower ratio, there is a large difference (+ or - 0.002) in impact.

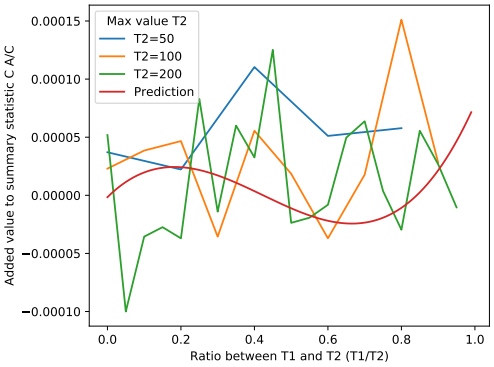
Repeating Optimizations All of these regressions have helped us define A_{T1}, A_{T1}, $A_{(C\to A)}$ and $A_{(C\to B)}$ in a more accurate way and we now know what has to be applied in every case. The only problem remaining is that the accuracy is not perfect due to the offsets, meaning that we cannot compute the perfect combination of inputs that need to change to find the best result in one step. The easiest solution that we use to solve this problem is to change inputs and look at how close we are to the summary statistics. If we are close enough (for example 0.01, which is the value that we use) to the summary statistics, then we can stop. If we are not close enough, we loop an extra time. By repeating this process until we have reached our wanted accuracy, we can find a solution in a small amount of steps. For our example of English, Dutch and French, we find a solution in three steps. A snippet of the output is given. In this snippet, we can see on the



(a) Change C to B



(b) Change C to B

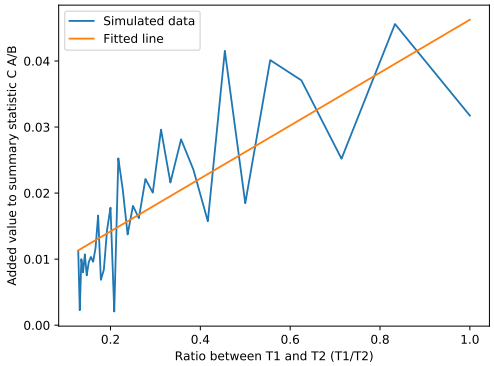


(c) Change C to B

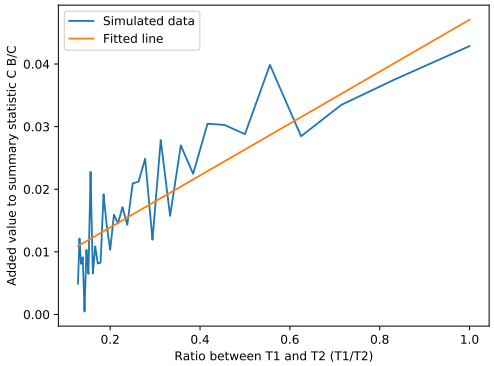
Figure 3.7: Solving the Second Problem: Increment for summary statistics, given an increase of proportion of 0.01 in the admixture event C to B. Average results for 200 runs. Here again, at a lower ratio, there is a large difference (+ or - 0.002) in impact for the figure 3.7a and 3.7b. Even for a higher ratio, the blue line is off the predicted line if we look at figure 3.7b.

left the values that need to be reached and on the right the values from only the T1 and T2 guess that have been found initially. This will help us to find out what to change so we can decrease the summary statistic that represents the difference between C and A by around 0.065, while decreasing the value of A and B by 0.05. This will be changed in the next step by adding 0.147 proportion to the C to A admixture event, together with a decrease of T1 and a slight increase of T2. We repeat this process, because we decreased the difference between C and A too much, and we decreased the difference between A and B too little. After three steps, we reach summary statistics that are very close to the ones of the real data. This indicates that the program is done and the results are written to a `.csv` file for analysis. One line in this `.csv` file contains the three languages, the population split times and the proportions for the two admixture events.

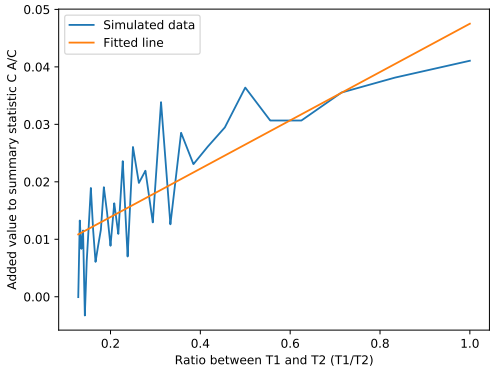
Listing 3.6: Output of execution for languages Dutch English and French



(a) Change in T1



(b) Change in T2



(c) Change in T2

Figure 3.8: Increment for summary statistics, given an increase of 5 generations (T1 + 5 or T2 + 5) in the population split events. Average results for 200 runs.

```
41
131
0.35602094240837695 - 0.40569999999999995
0.7277486910994765 - 0.728675
0.6649214659685864 - 0.730675
{'c_to_b': 5.363445591190272e-07, 'c_to_a': 0.1467787452967679}
35.28613973800967
131.87981597491589
0.35602094240837695 - 0.37045
0.7277486910994765 - 0.73285
0.6649214659685864 - 0.6530750000000001
{'c_to_b': 0.002839456796590863, 'c_to_a': 0.11129602155097616}
34.52493187266751
130.4526828994037
```

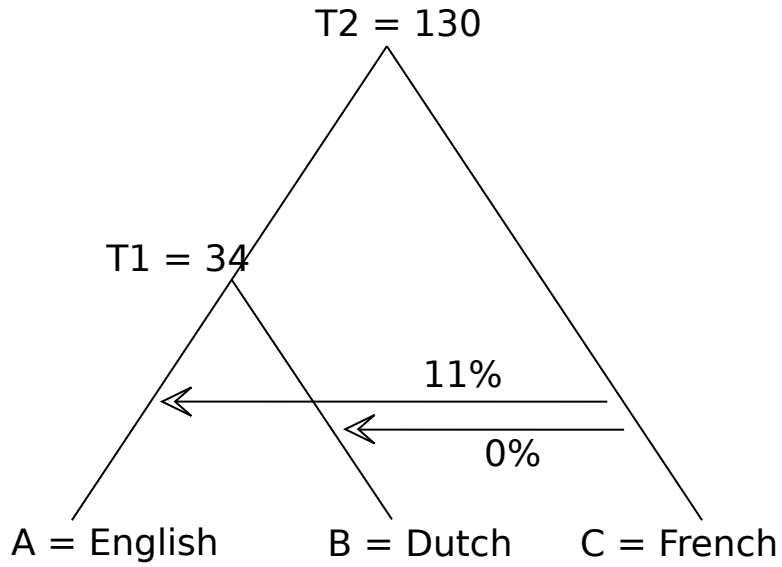


Figure 3.9: Language tree with split values and admixture proportions filled in. We see that the value of T1 is 34 generations ago and that the value of T2 is 130 generations ago. We see an admixture event that has a proportion of 0.11 or 11% from French to English.

```
0.35602094240837695 - 0.3517
0.7277486910994765 - 0.723775
0.6649214659685864 - 0.6641499999999999
{'c_to_b': 0.002839456796590863, 'c_to_a': 0.11129602155097616}
34.52493187266751
130.4526828994037
Done!
```

In figure 3.9, we can see the values filled in the language tree so that we get a final tree. We can see that French has some influence on English, while having almost no influence on Dutch. We can see that Dutch is close to English thanks to T1 and that the most recent common ancestor of French, English and Dutch can be found 130 generations ago.

Computing Final Input Parameters Three trees can be computed for each combination of three languages. This resulting data can be aggregated to see some broader patterns. Of course, not every combination will make sense, which is the reason we sometimes have false trees as a result of this process. False trees are trees that do not reach an accurate representation of the summary statistics of the real data. These trees will not be used in the next steps. False trees can be detected in two ways. The first way to notice that a tree is false is when one of the admixture rates has a proportion of more than 99%. This means that the language that is in the spot of language C as shown on figure 3.1, should be swap its spot with either language A or language B. The language that it needs to swap with the language that has the smallest proportion of the two. The code below shows what the output will look like.

Listing 3.7: Output for a false tree

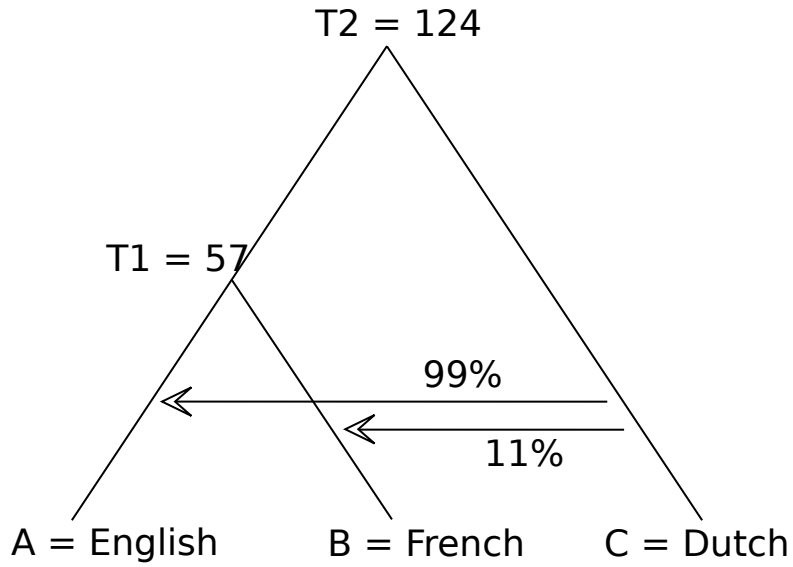


Figure 3.10: False language tree with split values and admixture proportions filled in. We see that the value of T1 is 57 generations ago and that the value of T2 is 124 generations ago. We see an admixture event that has a proportion of 0.11 or 11% from Dutch to French and 99% from Dutch to English.

```
Input Languages: ['English', 'French', 'Dutch']
...
{'c_to_b': 0.11519398016258979, 'c_to_a': 0.9991160311339595}
57.62670029316164
124.41678639310061
Done!
```

Here, as can be seen on figure 3.10 we can see that `c_to_a` indicates that the tree is wrong, so we should swap C (Dutch) with B (French). There is a second possibility for a tree that is wrong and those are the trees where the languages are closer to each other. The population splits will then be very close to each other, in an attempt to make "T1" greater than "T2". One example of this phenomenon is the combination Bulgarian (A), Sorbian Upper (B) and Slovenian (C). The result can be seen on the tree on figure 3.10.

Listing 3.8: Output for a false tree

```
Input Languages: ['Bulgarian','Sorbian Upper','Slovenian']
...
{'c_to_b': 0.8028868928201442, 'c_to_a': 0.6722441885907366}
45.268289412921845
45.268386948992415
Done!
```

Both of these results can be filtered out, because they give false information that is detrimental to further analysis. These problems that trees can have are not mutually exclusive, they can

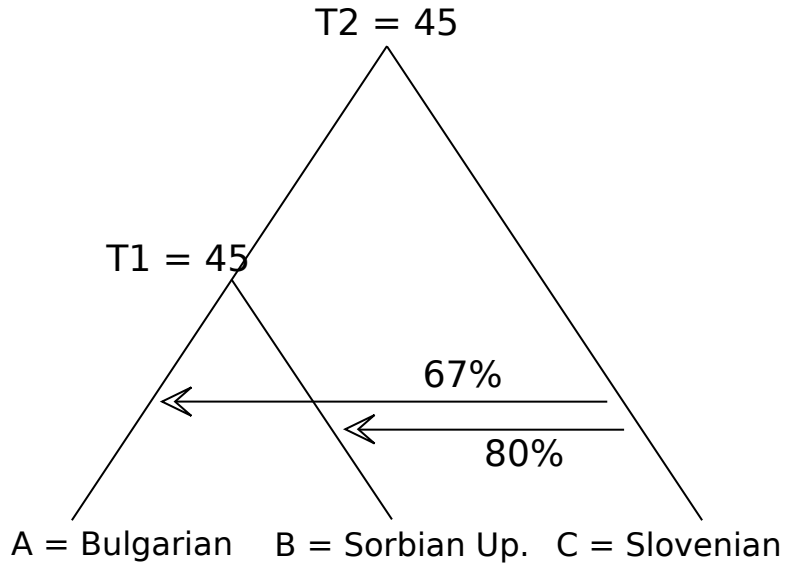


Figure 3.11: False language tree with split values and admixture proportions filled in. We see that the value of T1 is 45 generations ago and that the value of T2 is 45 generations ago. We see an admixture event that has a proportion of 0.80 or 80% from Slovenian to Sorbian Upper and 67% from Slovenian to Bulgarian.

both appear at the same time. A general trend can be extrapolated from this whole section. Examples of what to expect from the analysis are given in table 3.4.

3.4 Using Data From Experiments

We have gathered language data using language triplets from the procedure explained in the previous sections, but we still do not have a broader picture of the ancestries and exchanges that happen between more than three languages. In this section we will explain what has to be done in order to make sense of the bigger picture. We will also explain the process that has to be followed in order to determine the borrowings and the structure of language groups and families based only on the language triplets and inputs that have been computed for these triplets using coalescent simulations.

3.4.1 Computing Horizontal Exchanges Between Languages

When we look for horizontal exchanges, we look for languages that are close to each other, without looking at any ancestral relationship (parent, child, sibling). We will look at borrowings to determine whether there exists exchange between languages. We want to know what language is close to another, without trying to link them together to form a language tree for all languages. The result should look like a graph, where where every node $v \in \mathbf{V}$ represents a language, every edge $e \in \mathbf{E}$ is directed so that $e(v_{L1}, v_{L2})$ represent the exchange from language $L1$ to language $L2$.

How do we calculate a value for the borrowing from the data that we got from the previous

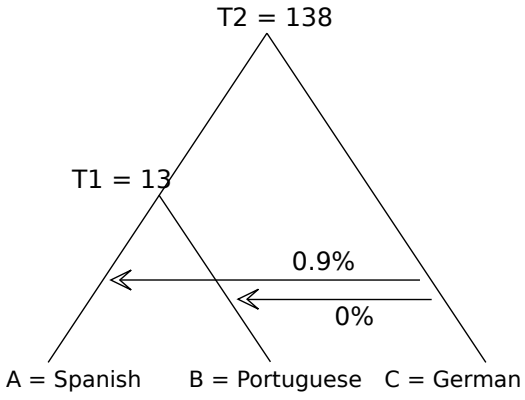
AB	BC	AC	Output prediction
0.5	0.7	0.7	A & B are similar, C is different, no borrowing needed to explain
0.3	0.5	0.7	A & B are similar, C is more similar to B than A, admixture needed from C to B
0.3	0.7	0.5	A & B are similar, C is more similar to A than B, admixture needed from C to A
0.5	0.5	0.5	All languages are similar, nothing needed
0.3	0.3	0.7	Unlikely, A is similar to B and B is similar to C, A must be similar to C
0.3	0.7	0.3	Unlikely, A is similar to B and A is similar to C, B must be similar to C
0.7	0.5	0.5	Unfeasible, A and B should be at least less different than B and C or A and C

Table 3.4: Table that shows what predictions we can make for the values for admixture proportion by looking at the summary statistics of the real-world data. Example numbers are given but the relation between the numbers is the most important part.

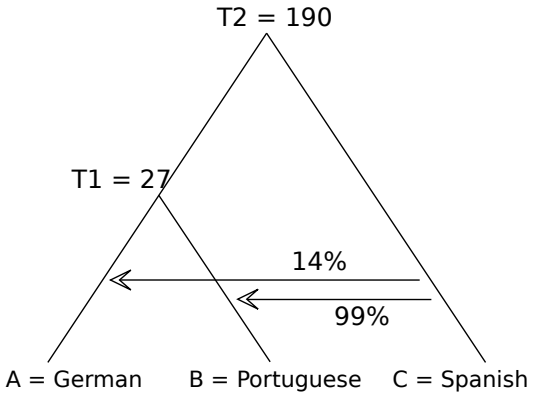
step? We will only look at the borrowings given by the admixture rate proportion previously computed between the languages. In other words, our approach uses the (C →A) and (C →B) values from the previous step. But there is a problem with that, that can be easily be shown with an example: take Spanish, Portuguese and German, we received three values shown on figure 3.12. If we filter out the wrong trees (see section 3.2.3), only one of these results remain, Spanish as language A, Portuguese as B and German as C. Although this can be helpful to know what influence German has on the two other languages, we still do not know how close or far Spanish is from Portuguese (we assume it must be close to each other, looking at the fact that it is the only tree that is not false and that the T1 value is low). The second smaller problem that has to be solved is the problem that a result with a high "T1" value, will subsequently cause the (C →A) and (C →B) to be higher (if there is any), because there was more time elapsed between 0 generations ago (present) and T1 generations ago. The solution to this smaller problem is to simply divide the value by the "T1", so we can get an average over time.

For the first problem, we have to look at other combinations of languages that can help us determine these values. An example could be to add Italian, and do the same procedure. This time, we can get information from trees that are not completely right, but cannot be classified as false trees. For example, you can have a tree where Italian is language A, Portuguese B, and Spanish C. In that case $Spanish \rightarrow Italian = 0.524$ and $Spanish \rightarrow Portuguese = 0.831$. We still have to divide that value by T1 in our computations. The main idea is to use these almost correct trees to compute numbers for languages we have no information about. The difference between those trees and the false trees, is that they have reached an accurate representation of the summary statistics, even if the languages are in the wrong position in the tree. The workflow is as follows:

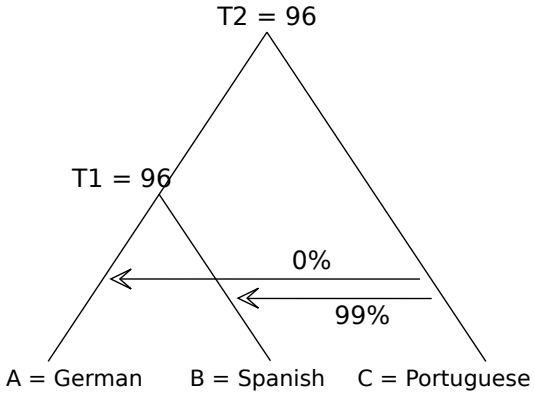
1. Filter out all the (C →A) and (C →B) values that are above 0.99.
2. Filter out all the (C →A) and (C →B) values that are equal to each other.
3. Create two new values that will be associated with each language triplet: $\frac{C \rightarrow A}{T1}$ and $\frac{C \rightarrow B}{T1}$.
4. Group up all nodes that have the same language in C and A or B and take the average.



(a) Combination 1



(b) Combination 2



(c) Combination 3

Figure 3.12: Experiments showing all results for all combinations of the languages: Spanish, Portuguese and German. Only one of these trees is valid, which is the tree shown on figure 3.12a.

For example, take all the rows with French as language C and Spanish as language A and take the average of the value from the previous step.

5. Determine two values: one value will be the strong connections and the other will be for weak connections. This can be done with the help of quantiles or percentiles. In our study, we use 90% percentile and 70% percentile, but this can be chosen freely depending on how grouped the nodes need to be (this will be explained later).

6. Remove every relationship that has a value smaller than the value given for the weak connections.

Now, we should have a number of pairs of two nodes (or languages), together with their respective average value of borrowing, we can create a graph to better visualize these numbers and to find groups that we will use in the next section. The graph will simply be a graph containing all the languages and we will place directed edges for values that are in one of the two

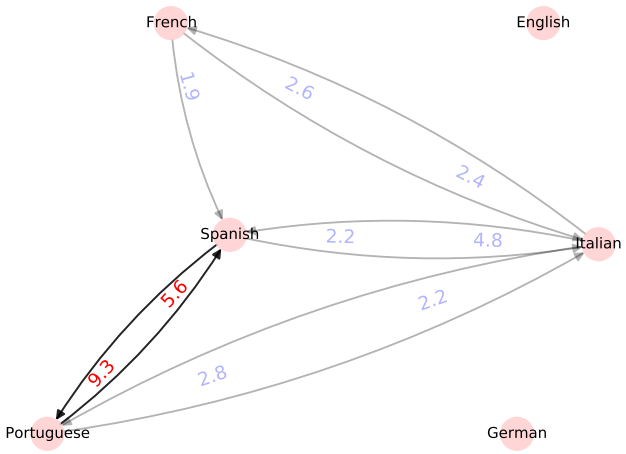


Figure 3.13: Graph of borrowings between a sample of Western European languages. Grey lines represent the weak connections, black lines represent strong connections. The number closest to the node is the number that represents the exchange FROM that node.

lists. At the end, we search for strongly connected components in the graph to form groups of languages. An example result is shown on figure 3.13. The numbers on figure 3.13 is the number that we have computed in the previous step. There is one strongly connected component in this example: Spanish and Portuguese.

3.4.2 Computing Vertical Ancestries Between Languages

In the previous section, we have seen how we got initial groups of languages that are very similar by using strongly connected components. We can work with these groups and try to group up groups of languages (or single languages) until all the nodes (languages or linguistic varieties) are connected to one another.

To do this, you have to compute the average distance between all the different groups. This is done with the help of T1.

As an example, imagine having three groups: Italian and French, Spanish and Portuguese and German and Dutch. In this case, if we want to compute the average distance between the first group and the second group, we check the average T1 if we put French as language A, Italian as language B and Spanish as language C. We repeat this and replace Spanish with Portuguese. We can do the same for the two other group combinations.

The two groups with the smallest average distance are combined and the process is repeated. Once there are only two groups left, we can combine these two groups. A graph can be created representing this process, where every two groups are combined into a parent group, which receives the value of the average distance. The figure 3.14 shows what the output would be for the graph shown on 3.13. In this figure, the node 128.5 is represents most recent common ancestor of all the languages in this graph. Parents of languages use the average T1 and parents of groups of languages or a combination of parents of groups and languages use the average T2. This is done because we have to use the values that we know from the language trees. In the language trees two languages combine into a parent node T1 generations ago, while T2

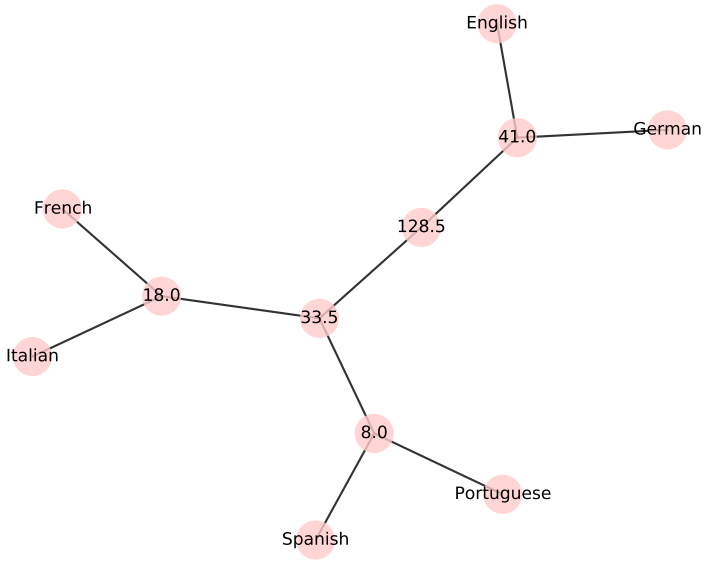


Figure 3.14: Graph of ancestry for a sample of Western European languages. Languages are grouped and into parent nodes which show the time in generations ago for the most recent common ancestor between the languages.

represents the combination of a language and the parent of a group of two languages. In this example Spanish and Portuguese need to be put under T1 in the language tree (same for French and Italian or German and English). Other more interesting examples will be discussed in the results section.

3.5 Training a Multi Output Regressor

Previously, we have seen how we slowly changed the input of simulations in order to get results that look like the summary statistics given by the real data. This process is done in two steps, first we compute some preliminary results and then we loop multiple times in order to be as accurate as possible. One could argue that this process is time-consuming, especially if a lot of linguistic varieties or a whole language family needs to be analyzed. As an alternative to this time-consuming process, we need to find a fast alternative that finds a set of inputs, given the three summary statistics.

For this purpose, we will train a Multi Output Regressor. The goal of this regressor is to find the input parameters for the simulation that will reach accurate summary statistics. The data it will train on is a set of computed inputs for 1092 combinations of languages that are found in the Slavic language family, together with some outliers, such as Italian and Greek. That data needs to be fed into a supervised learning regression algorithm. In our study, we use a random forest regressor for this task, because it provides us with a score around 0.85 and a good speed. This is not perfect, but we can expect the training data to have some irregularities too. We have to remember that the coalescent inherently is a method that uses randomness and that some variation in the results can happen. The other irregularity is the accuracy we put in the previous section, because we had to find a result that was ”good enough”. We give two examples

Inputs			Real Outputs				Predicted Outputs			
A/B	B/C	A/C	C →B	C →A	T1	T2	C →B	C →A	T1	T2
0.256	0.285	0.275	0	0.06	28	32	0.09	0.07	26	33
0.855	0.288	0.855	0.2	0.99	64	149	0.20	0.99	62	150

Table 3.5: Comparison of results given by the regressor compared to the results found by the classic method presented previously.

of predicted inputs to the model compared to inputs computed with the method previously explained. The results are shown on table 3.5. The outputs are not always similar, but they are similar enough to be used as a fast way to explore a lot of data (as indicated by the score of around 0.85). As an example of why we require speed sometimes: if we take all the combinations possible in the IELex database, we have to compute the inputs for 66300 simulations. If one of those computations takes approximately 30 seconds, it would take 23 days to get the full results for an Intel Core i7-9700k running in parallel on 8 cores. With this method, it takes 3152 seconds or 52 minutes and 32 seconds to predict the inputs of the simulations with the regressor.

The regressor used is available with scikit-learn (Pedregosa et al., 2011; Buitinck et al., 2013). The parameters of the regressor have been optimized using `GridSearchCV`, which is a method that tests multiple parameters and find which one performs the best. In our case with our data, we use 250 estimators, without any maximum depth. The maximum amount of features is found with the "auto" parameter.

As an experiment, we can look back at the graphs given before and compare the results we get from the regressor to the results we got from the analysis with the coalescent simulations. We can see that the regressor underestimates the borrowing a bit in this example, as shown on figure 3.15, but the ancestry is really close, as shown on figure 3.16.

3.6 Evaluating Vertical Exchanges

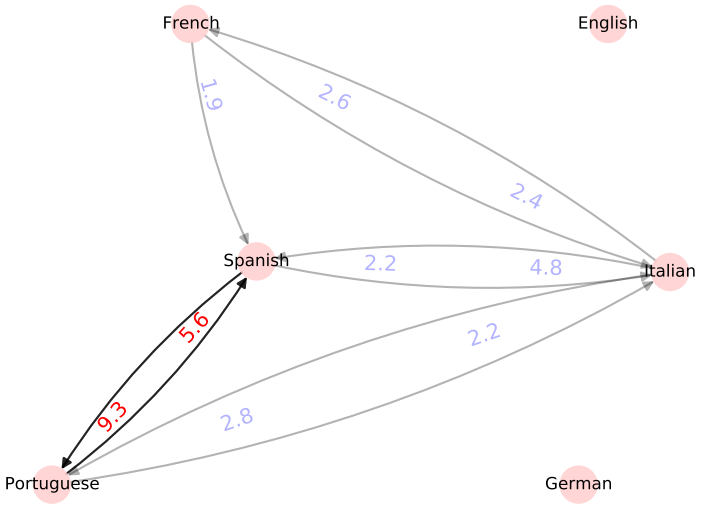
To evaluate the correctness of the vertical exchanges, we can simply look at the previously known Indo-European language family. The goal will be to find the distance between the real family shape and the shape found by our method. This can be done by computing the quartet distance. The quartet distance is a measure for distance between phylogenetic trees. A quartet distance of 0 means that the tree has the exact same shape. This technique has previously been used in Pompei et al. (2011).

As an example, we can try to compute the quartet distance of the three given on figure 3.14. If we compare this to the reference tree from Hammarström et al. (2021), we see that that there is a small difference with Italian. In that case Italian should be in a separate branch. The quartet distance in that case is 4.

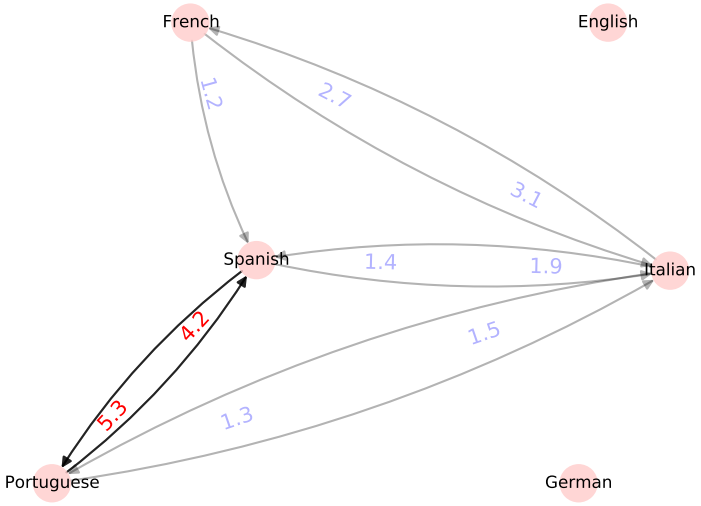
To compute this distance we will use the `tqDist` package in R (Sand et al., 2014) and compare it to the Indo-European tree from Hammarström et al. (2021). In cases where our tree has some languages that are not in the reference tree, we will try to compare more manually.

3.7 Evaluating Horizontal Exchanges

Evaluating horizontal exchanges is a bit more tricky. In our case, we want to see if our found borrowing measures corresponds to the reality. The particularity about the IELex database is that it contains words from the Swadesh list, which are words that are not often borrowed.



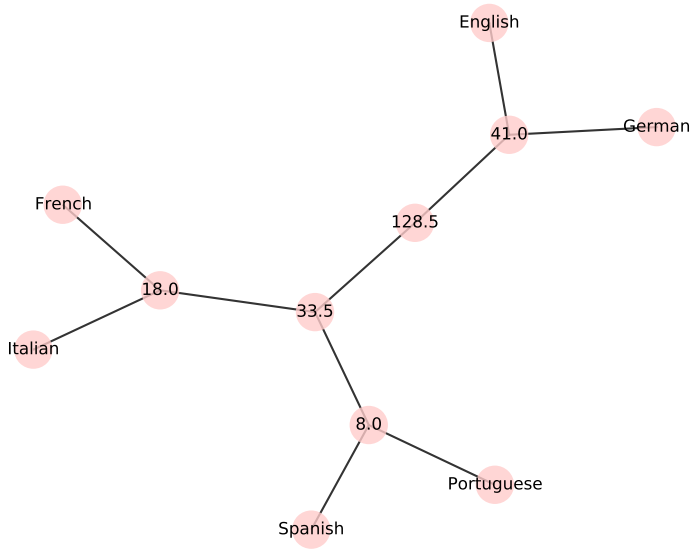
(a) Prediction with coalescent



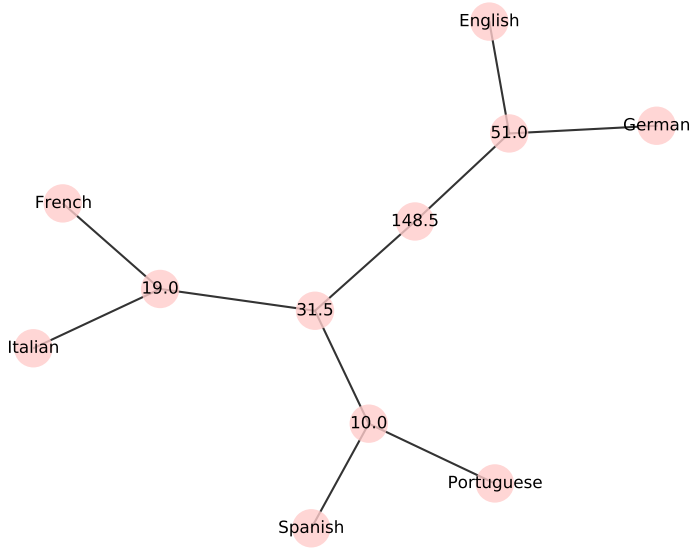
(b) Prediction with regressor

Figure 3.15: Prediction for the borrowing graph on the right, compared to prediction done with the coalescent on the left. The values are off by some units, but the structure of the graph is intact. The strong links are also the same in the prediction. This is the strong link between Spanish and Portuguese.

Additionally the database that we have is not annotated with borrowing data. This makes it difficult to retrieve the percentage of words that are borrowed from another language.



(a) Prediction with coalescent



(b) Prediction with regressor

Figure 3.16: Prediction for the ancestry graph on the right, compared to prediction done with the coalescent on the left. Some values are overestimated and some are underestimated, but generally the same structure is kept. The biggest difference is the root, which differs 20 generations from the value we found with the regressor.

We will use expert annotated Indo-European linguistic data from List, Nelson-Sathi, et al. (2014) in order to make a case study on borrowing. This data indicates if a word is loaned, but

not what language it is loaned from. This means that we cannot know from which language a word is borrowed. In this data, some languages have no loan words. What we will do is look at the overall share of the language that is borrowed and compare it to our model. To find this share of the language that is borrowed, we will sum all the weaker links of the borrowing. We avoid taking stronger links, because they have a strong influence of vertical exchange.

This is also the difficulty with horizontal exchange: the strongest links between the languages exist because of vertical exchange. For example, if we look at any loanword database, such as the World Loanword Database (WOLD), we will find no loanwords from Spanish to Portuguese (or the inverse), although the languages are similar. This is because most of the similarities are explained by vertical exchange (a previous common ancestor). Our horizontal exchange results show strong influence from the vertical exchange.

Chapter 4

Results

In this final section, we will be discussing examples of language families that we can model with the help of our method that we developed in the previous sections.

4.1 Basic Case: Sample of Two Language Families

In this case, we discuss what happens in a simple case, where we chose a small number of languages in a certain family A and another small sample from a family B. In this example A will be Romance languages and B will be Germanic languages. This example is the same as the one seen previously, but this time we will discuss exactly what we can see in figure 3.13 and 3.14. We have applied our method on these languages and as we can see on figure 3.13, there is a strongly connected component between Spanish and Portuguese. This component forms a subgroup on figure 3.14, and as we can see the most recent common ancestor is only 8.0 generations ago. We do not see much exchange between the two families. Although there is an exchange from German to English if we lower the percentile for weak connections. This can be seen on figure 4.1. Code snippet 3.6 showed the result that French had a proportion of 0.11 for the admixture event. This does not appear in the borrowing graph, because we have to divide 0.11 by 34, which makes it have a smaller impact than first observed. This just means that we can adapt the visualization by changing the thresholds for stronger and weaker borrowings. We can conclude out of this experiment that there is not a lot of exchange between Germanic languages and Romance languages. We can also understand how relatively strong connections are, just by comparing the values shown. About 28.30% of all the 80000 English words analyzed in Finkenstaedt & Wolff (1973) came from French, which does not seem to correspond to the results we concluded. Another large part of the words come from Latin. We expected more borrowing in our results, but it could be that the words from the Swadesh list are so basic that they are specifically resistant to borrowing, which makes our program underestimate the borrowing.

4.2 High exchange case: one language family

A good example of languages where there is a high amount of exchange are the languages that are found in the Balkans, a region in Southeastern Europe. There are many languages spoken there, but we will only restrict ourselves to the data that we have, that is to say Indo-European languages. This excludes for example Hungarian, because it is not a Indo-European language.

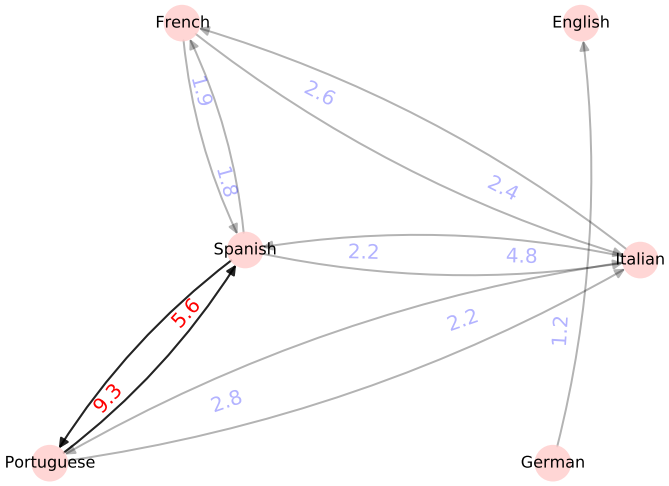


Figure 4.1: Graph of borrowings between a sample of Western European languages. A lower threshold for weaker connections make new connections visible.

We apply our approach on this set of languages to find out what we can discover. Figures 4.2 and 4.3 show the results for this example. We can see that here, most of the nodes are connected with a stronger connection than in the previous example. Greek, even though it is a language close to the Balkans, has no influence on any of the languages. Bulgarian seems to be the language with the most influence on the languages of the Balkans. If we look at the ancestry, we can see that two groups have formed. One group contains Bulgarian and Macedonian, which are the Eastern South Slavic languages, and the other group contains Serbo-Croatian and Slovenian, otherwise known as the Western South Slavic languages. Of course, we have to use a combination of both graphs to get the full picture of the situation. Each graph gives extra information: the ancestry graph does not show for example that Bulgarian has a lot more influence than Macedonian on the languages of the Balkan. We can build on this example further into the whole Slavic language family.

We can try to evaluate this subset of languages by using the quartet distance. Here the quartet distance is equal to 0, which means that we have a perfect tree.

4.3 Full family with outliers

We have seen some more restricted examples previously, but not what it looks like for a full language family. To be able to compare to previous results, we are going to take the Slavic language family as an example, together with some outliers. This will help us to understand what the models can say when the languages are not from neighboring countries. Figures 4.4 and 4.5 are interesting to compare to previous results. In the borrowing graph for Slavic languages, we can see the clusters that form the areal groups: Ukrainian and Belarusian (East Slavic), Sorbian Lower and Sorbian Upper (Sorbian), Czech and Slovak (Czech–Slovak), Bulgarian and Macedonian (Eastern South Slavic) and Slovenian with Serbo-Croatian (Western South Slavic). These clusters can be found again in the ancestry figure 4.5, where we can see how they link

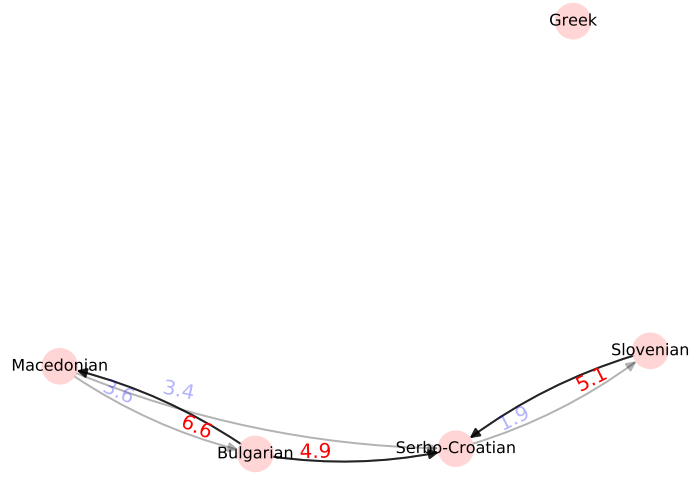


Figure 4.2: Graph of borrowings between the languages spoken in the Balkans. Greek clearly is not in the same category as the other languages in the Balkans. The links between the languages in the Balkans are stronger than in the previous example of Western European language.

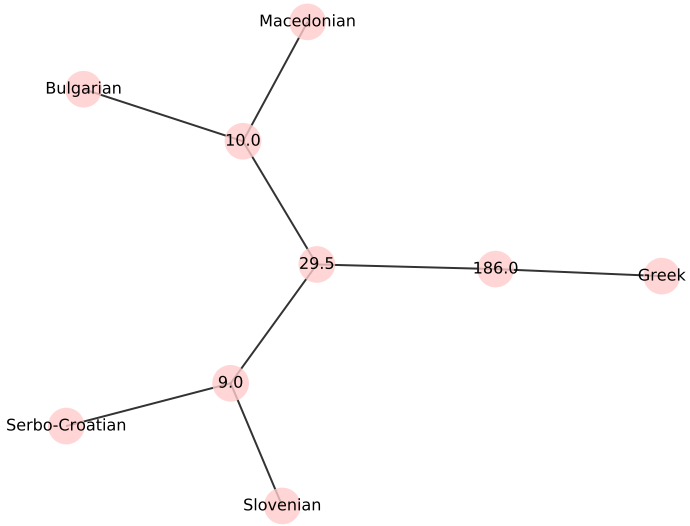


Figure 4.3: Graph of ancestry between the languages spoken in the Balkans. Most of the languages are relatively close the each other, except for Greek. The most recent common ancestor between the languages of the Balkans and Greek is 186.0 generations ago, which is far away from the smaller numbers we can find in the other languages of the Balkans.

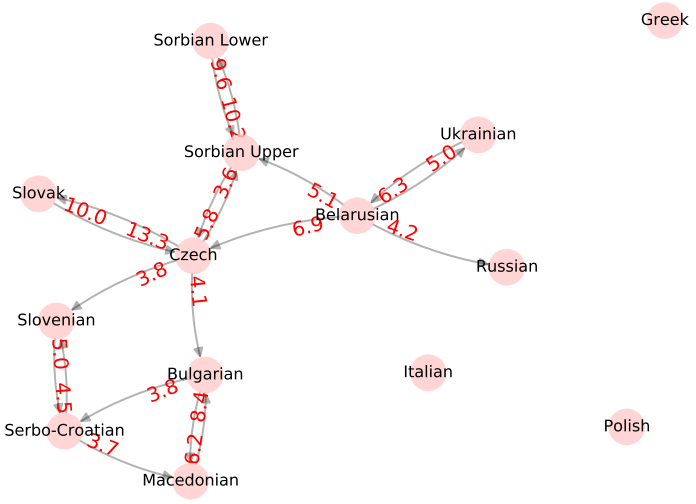


Figure 4.4: Graph of borrowings between the Slavic languages and some outliers. Some links from the languages of the Balkans seem stronger here than previously shown. This happens because we collected more data that can explain how big the influence is of one language to another. We have removed the weaker links for clarity purposes.

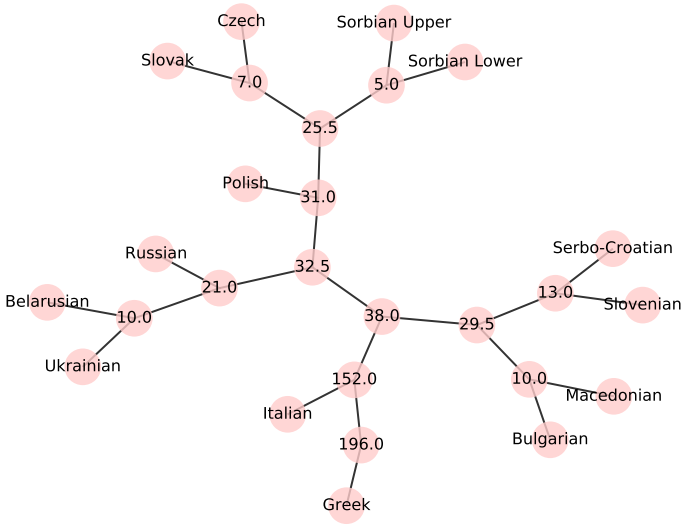


Figure 4.5: Graph of ancestry between the Slavic languages and some outliers.

together with the languages that were not mentioned previously. There are three historical families we can find in this figure: West Slavic, South Slavic and East Slavic languages. For example Polish joins Czech-Slovak and Sorbian to form the West Slavic languages. This also tells us that the West Slavic languages seem to be closer to the East Slavic languages than the South Slavic languages. This relatedness seems to be an artefact of the fact that they are closer to each other geographically speaking.

If we evaluate the tree in with quartet distance, we have a distance of 165. This seems to suggest that our tree does not fit at all, but if we analyze the problem, we immediately understand where the problem lies. In general, our trees are binary, except for the initial groups. In the real-world data, the groups are not binary. For example, to get a perfect match, Polish should join the node 25.5 on figure 4.5. All the initial groups match with the real-world data, except for Russian which should be under the node 10.0.

4.4 Predicted Indo-European Languages

We already previously explained how it was too computationally expensive to run the program for many languages. This case will test if the regressor that we have built is strong enough to build a model that is clear and true for the whole IELex database. This includes languages from many different backgrounds, dead languages and languages spoken by a very small population. This whole computation was done with the previously explained regressor, some values may be a bit off (times may be a couple of units off), but the general idea is to get a quick grasp of the whole Indo-European language family. As we can see in figure 4.7, to root is located at node with time 196.0. The nodes are colored with the ”modularity” tool from Gephi. This means that these are the found clusters inside the tree. We can see that the regressor has had some trouble with the values that are closer to each other, like seen with the Slavic languages. The borrowing graph shown in figure 4.6 is a bit more cluttered. Here, we color the clusters again to get a good idea of the different groups of languages. The idea here is not to look for families, but to find strongly connected components.

Some interesting findings can be found in this figure. The line width shows how strong the connection is. First, we can see that there are eight groups in the Indo-European languages, with some of the groups being larger than others. There are also some other languages that are not part of a strongly connected group. We can find the Italic languages, which is the group of languages that include Latin, and the Romance languages. A strong connection exists between Italian and French, and between Portuguese, Spanish and Catalan. Latin has weaker connections to the languages of this group, as it is only linked to Italian. A second group we can find are the Germanic languages. They include the North Germanic languages and the West Germanic languages, along with some older varieties of these languages. There are smaller groups that are formed only with the older variety of the same language. For example, Ancient Greek and Greek and Old Irish and Irish in smaller groups. The Armenic group of languages exists in the same way that Greek is one of the branches of the Indo-European language family, but they . Irish and Old Irish are separate from the group of Breton, but in the family, they should be under the same group called the Celtic languages. It could be that the they are too different from each other to be placed in the same group. The other large group we can find are the Balto-Slavic languages. Many of the languages in this group are strongly connected to each other, indicating that there is a lot of exchange between these languages. Another smaller group that can be found is the group that forms the Indo-Iranian languages, which has six members. Unfortunately, our program cannot detect the connection between these languages. The links between these languages are relatively weak compared to other groups. The last group is the

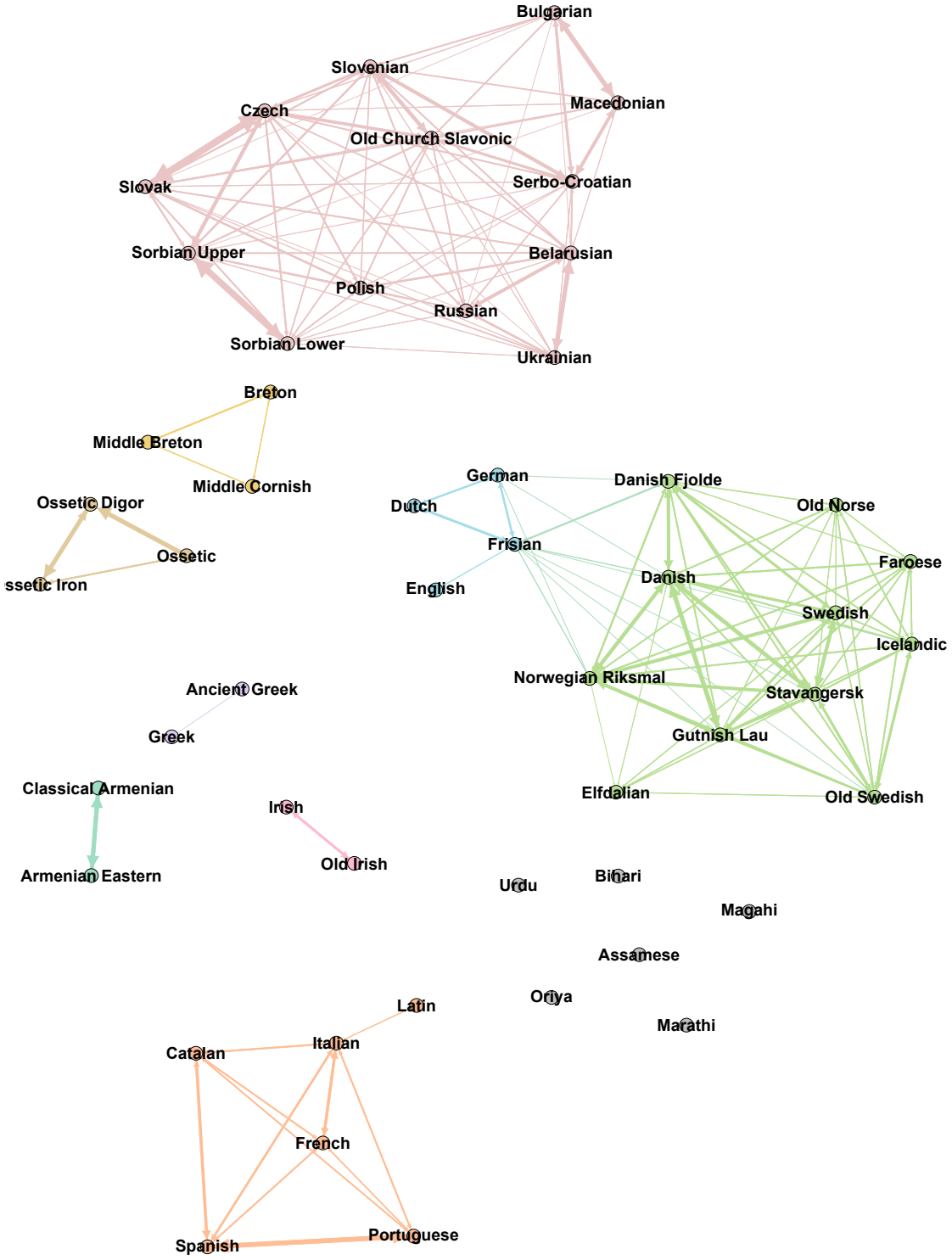


Figure 4.6: Graph of borrowings between all the Indo-European languages from IELex. Indo-Aryan (Grey), Brythonic (Ochre), Goidelic (Pink), Italic (Orange), Ossetic (Light brown), North Germanic (Dark green), Blue (West Germanic), Balto-Slavic (Dusty grey), Greek (Light grey), Armenian (Dark cyan).

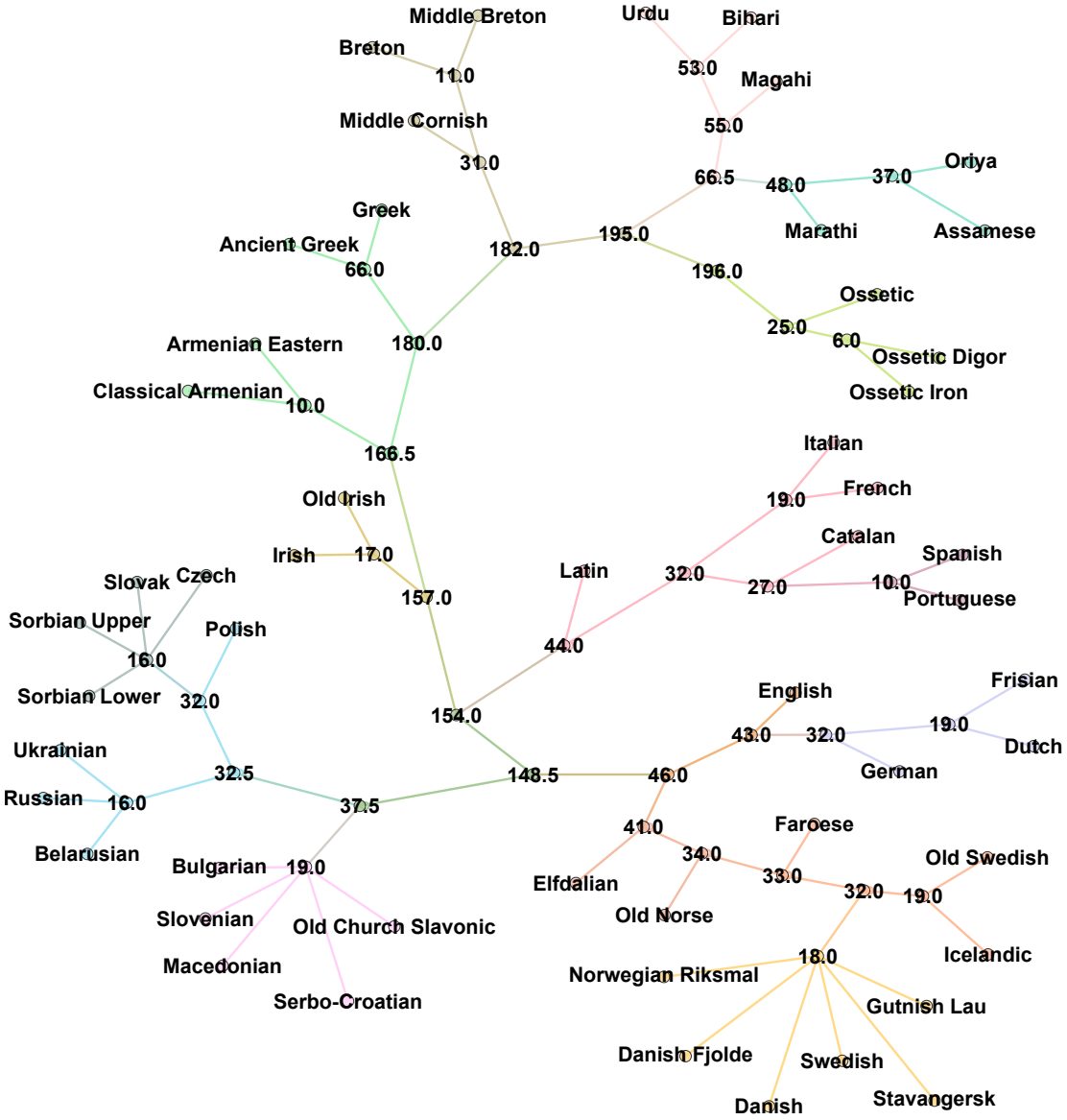


Figure 4.7: Graph of ancestry between all the Indo-European languages.

Ossetic group. This group is part of the Iranian languages under the Indo-Iranian group of languages. They differ enough from the previous group that they are not grouped together. This can also be seen in the graph shown on figure 4.7. The names for the groups inside the Indo-European language family can be found in Hammarström et al. (2021).

Again, our binary tree restriction here makes it more difficult to evaluate if this tree corresponds to the Indo-European language family tree we know of. Additionally, this tree has been calculated by the regressor, so we expect errors that the program would have not made. Last but not least, many smaller or dead languages have a limited amount of concepts defined compared to the more popular languages such as English or French. Additionally, some languages are not in the tree given by Hammarström et al. (2021), such as Stavangersk, which means that we cannot use the quartet distance in this case.

Instead, to evaluate this tree, we will look at the general structure of the branches. All of the sub branches of the tree correspond to the ones that can be found in the Indo-European language family tree. There are two error in our generated tree, according to the tree found in Hammarström et al. (2021). French should be attached closer to Spanish, Catalan and Portuguese than Italian. For the North Germanic languages, there could be more distinction between the groups. For example, we should see a more tight group between Faroese, Icelandic and Old Norse.

Here, we will also evaluate the borrowing. By summing the weaker links, we try to find what languages have had the most borrowed lexicon and compare it to the known percentage borrowed (of the Swadesh list). As can be seen in the table 4.1, some values correspond, while some other values are completely off. The most problematic values are Elfdalian, Faroese, German, Macedonian and Urdu. The other 18 languages have similar values to the real ones. We have to remember that this is the data that is found with the regressor, so we expect some inaccuracies for some values.

	% borrowed	
Language	Real	Model
Assamese	7.6	3.9
Bihari	7.9	6.2
Catalan	0.1	1.1
Czech	0.4	1.3
Danish	1.5	3.2
Elfdalian	0.4	9.2
English	13.4	15.3
Faroese	0.3	5.2
French	1.4	1.1
Frisian	1.8	4.5
German	2.3	11.0
Macedonian	0.5	4.9
Magahi	6.3	8.1
Marathi	7.6	5.6
Oriya	7.1	5.1
Ossetic	1.6	2.5
Polish	0.9	2.3
Russian	0.9	2.6
Slovak	0.4	4.0
Spanish	1.5	1.2
Swedish	2.0	4.0
Ukrainian	1.3	2.6
Urdu	15.5	5.2

Table 4.1: Percentage of borrowed words in IELex database compared to the percentage of words borrowed in the model.

Chapter 5

Conclusion

We have researched multiple different techniques to look for borrowing and ancestry inside a language family. Many techniques relied on manual historical linguistic work, genetic data or lexical, morphological and phonological data, which can be hard to acquire. We propose a technique to find these exchanges by using the coalescent theory, a theory often used in the field of population genetics. It allowed us to build trees using cognate data from the IELex database. Our technique allows us to find exchanges by adapting inputs of the model in order to get as close as possible to the real cognate data. The optimization method used to get as close as possible to the summary statistics of the real data relied on minimizing the differences between the summary statistic given a set of equations. Using these results, we have found that it accurately represented the language families, by comparing the groups found to the groups that we know of today. We also created a regressor that can analyse all the data from the IELex database in under an hour on a single machine, which provides us with a faster analysis than the research we have found.

For a long time, the coalescent theory has exclusively been used in population genetics. This research shows that the coalescent is a technique that can be used by researchers in the field of computational linguistics. We hope that there is more interest in using this technique in the linguistic field in the future, especially in (computation) historical linguistics. In our case, it provided us a with a fast and accurate method to do linguistic phylogeny, only using cognate data. The analysis of exchange between languages seems a bit problematic, as it seems to be underestimate some of the influences, such as the influence between French and English. The problem seems to be that the vertical exchanges are difficult to separate from the horizontal exchanges.

5.1 Future Work

The method could be refined by adding known historical events in the model in order to find more precise data on the languages involved in this historical event. An example could be a migration from one country to another or a colonization.

Other case studies could be done specifically to allow us to understand some situations of language exchange better, especially between language families. The assumption of a most recent common ancestor could be tested thanks to this method.

As we have seen, the only data used in this case were cognates, but it could be an interesting study to see what differences we can find if we do not exclusively look at cognates. We could also analyse to see which features have to most relevance in research by looking at what impact

is created by changing the inputs of the coalescent simulation. For example, we could image a case study that researches syntactic data. The concepts in the data could also be separated into grammatical categories, for example pronouns, verbs. We could also imagine a division of the data based on topic categories, such as weather, animals or plants. This could create more summary statistics that could increase the accuracy of the model, and look at where the exchange happens precisely.

Finally, the findings can be researched to look for a relationship between the geographical distances and exchange.

References

Atkinson, Q. D. (2011). Phonemic diversity supports a serial founder effect model of language expansion from africa. *Science*, *332*(6027), 346–349.

Atkinson, Q. D., & Gray, R. D. (2005). Curious parallels and curious connections—phylogenetic thinking in biology and historical linguistics. *Systematic Biology*, *54*(4), 513–526.

Beerli, P., & Felsenstein, J. (2001). Maximum likelihood estimation of a migration matrix and effective population sizes in n subpopulations by using a coalescent approach. *Proceedings of the National Academy of Sciences*, *98*(8), 4563–4568.

Bouckaert, R., Lemey, P., Dunn, M., Greenhill, S. J., Alekseyenko, A. V., Drummond, A. J., … Atkinson, Q. D. (2012). Mapping the origins and expansion of the indo-european language family. *Science*, *337*(6097), 957–960.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., … Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *Ecml pkdd workshop: Languages for data mining and machine learning* (pp. 108–122).

Campbell, L. (2013). *Historical linguistics*. Edinburgh University Press.

Creanza, N., Ruhlen, M., Pemberton, T. J., Rosenberg, N. A., Feldman, M. W., & Ramachandran, S. (2015). A comparison of worldwide phonemic and genetic variation in human populations. *Proceedings of the National Academy of Sciences*, *112*(5), 1265–1272.

Crystal, D. (2008). *A dictionary of linguistics and phonetics*. John Wiley & Sons, Ltd. doi: https://doi.org/10.1002/9781444302776.ch3

Darwin, C. (2009). *The descent of man and selection in relation to sex* (Vol. 1). Cambridge University Press. doi: 10.1017/CBO9780511703829

Daumé III, H. (2009). Non-parametric Bayesian areal linguistics. In *Proceedings of human language technologies: The 2009 annual conference of the north American chapter of the association for computational linguistics* (pp. 593–601). Boulder, Colorado: Association for Computational Linguistics.

Drummond, A. J., Nicholls, G. K., Rodrigo, A. G., & Solomon, W. (2002). Estimating mutation parameters, population history and genealogy simultaneously from temporally spaced sequence data. *Genetics*, *161*(3), 1307–1320.

Dunn, M. (2012). *Indo-european lexical cognacy database (ielex)*. Nimegen: Max Planck Institute for Psycholinguistics. Retrieved from `http://ielex.mpi.nl/`

Finkenstaedt, T., & Wolff, D. (1973). *Ordered profusion; studies in dictionaries and the english lexicon* (Vol. 13). C. Winter.

Gong, T. (2010). Exploring the roles of horizontal, vertical, and oblique transmissions in language evolution. *Adaptive Behavior*, *18*(3-4), 356-376. Retrieved from `https://doi.org/10.1177/1059712310377241` doi: 10.1177/1059712310377241

Gray, R. D., & Atkinson, Q. D. (2003). Language-tree divergence times support the anatolian theory of indo-european origin. *Nature*, *426*(6965), 435–439.

Griffiths, R. C., & Marjoram, P. (1996). Ancestral inference from samples of dna sequences with recombination. *Journal of Computational Biology*, *3*(4), 479–502.

Hammarström, H., Forkel, R., Haspelmath, M., & Bank, S. (2021). *Glottolog 4.4*. Leipzig. Max Planck Institute for Evolutionary Anthropology. Retrieved from `https://glottolog.org/accessed2021-05-17` doi: 10.5281/zenodo.4761960

Hudson, R. R. (2002). Generating samples under a wright–fisher neutral model of genetic variation. *Bioinformatics*, *18*(2), 337–338.

Hudson, R. R., & Kaplan, N. L. (1988). The coalescent process in models with selection and recombination. *Genetics*, *120*(3), 831–840.

Hudson, R. R., et al. (1990). Gene genealogies and the coalescent process. *Oxford Surveys in Evolutionary Biology*, *7*(1), 44.

Hunley, K., Dunn, M., Lindström, E., Reesink, G., Terrill, A., Healy, M. E., … Friedlaender, J. S. (2008). Genetic and linguistic coevolution in northern island melanesia. *PLoS Genetics*, *4*(10), e1000239.

Hunley, K., & Long, J. C. (2005). Gene flow across linguistic boundaries in native north american populations. *Proceedings of the National Academy of Sciences*, *102*(5), 1312–1317.

Jacques, G., & List, J.-M. (2019). Save the trees: Why we need tree models in linguistic reconstruction (and when we should apply them). *Journal of historical linguistics*, *9*(1), 128–167.

Kaiping, G. A., & Klamer, M. (2018). Lexirumah: An online lexical database of the lesser sunda islands. *PloS One*, *13*(10), e0205250.

Kelleher, J., Etheridge, A. M., & McVean, G. (2016). Efficient coalescent simulation and genealogical analysis for large sample sizes. *PLoS Computational Biology*, *12*(5), e1004842.

Kingman, J. F. C. (1982). The coalescent. *Stochastic Processes and Their Applications*, *13*(3), 235–248.

Kuhner, M. K., Yamato, J., & Felsenstein, J. (1998). Maximum likelihood estimation of population growth rates based on the coalescent. *Genetics*, *149*(1), 429–434.

Kuhner, M. K., Yamato, J., & Felsenstein, J. (2000). Maximum likelihood estimation of recombination rates from population data. *Genetics*, *156*(3), 1393–1401.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (Vol. 10, pp. 707–710).

List, J.-M. (2014). *Sequence comparison in historical linguistics*. Düsseldorf: Düsseldorf University Press. Retrieved from `https://doi.org/10.5281/zenodo.11879` doi: 10.5281/zenodo.11879

List, J.-M., Nelson-Sathi, S., Geisler, H., & Martin, W. (2014). Networks of lexical borrowing and lateral gene transfer in language and genome evolution. *Bioessays*, *36*, 141 - 150.

List, J.-M., Shijulal, N.-S., Martin, W., & Geisler, H. (2014). Using phylogenetic networks to model chinese dialect history. In *Quantifying language dynamics* (pp. 125–154). Brill.

Liu, L., Yu, L., & Edwards, S. V. (2010). A maximum pseudo-likelihood approach for estimating species trees under the coalescent model. *BMC Evolutionary Biology*, *10*(1), 1–18.

Monaghan, M. T., Wild, R., Elliot, M., Fujisawa, T., Balke, M., Inward, D. J., … Vogler, A. P. (2009). Accelerated Species Inventory on Madagascar Using Coalescent-Based Models of Species Delineation. *Systematic Biology*, *58*(3), 298-311. Retrieved from `https://doi.org/10.1093/sysbio/syp027` doi: 10.1093/sysbio/syp027

Morlon, H., Potts, M. D., & Plotkin, J. B. (2010). Inferring the dynamics of diversification: a coalescent approach. *PLoS Biol*, *8*(9), e1000493.

Moss, G. (1992). Cognate recognition: Its importance in the teaching of esp reading courses to spanish speakers. *English for Specific Purposes*, *11*(2), 141–158.

Nelson-Sathi, S., List, J.-M., Geisler, H., Fangerau, H., Gray, R. D., Martin, W., & Dagan, T. (2011). Networks uncover hidden lexical borrowing in indo-european language evolution. *Proceedings of the Royal Society B: Biological Sciences*, *278*(1713), 1794–1803.

Nordborg, M. (2004). Coalescent theory. In *Handbook of statistical genetics* (chap. 20). American Cancer Society. Retrieved from `https://onlinelibrary.wiley.com/doi/abs/10.1002/0470022620.bbc21` doi: https://doi.org/10.1002/0470022620.bbc21

Palstra, F. P., Heyer, E., & Austerlitz, F. (2015). Statistical inference on genetic data reveals the complex demographic history of human populations in central asia. *Molecular Biology and Evolution*, *32*(6), 1411–1424.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pompei, S., Loreto, V., & Tria, F. (2011). On the accuracy of language trees. *PloS one*, *6*(6), e20109.

Powell, M. J. D. (1978). A fast algorithm for nonlinearly constrained optimization calculations. In G. A. Watson (Ed.), *Numerical analysis* (pp. 144–157). Berlin, Heidelberg: Springer Berlin Heidelberg.

Pybus, O. G., Charleston, M. A., Gupta, S., Rambaut, A., Holmes, E. C., & Harvey, P. H. (2001). The epidemic behavior of the hepatitis C virus. *Science*, *292*(5525), 2323–2325. Retrieved from `https://science.sciencemag.org/content/292/5525/2323` doi: 10.1126/science.1058321

Rama, T., List, J., Wahle, J., & Jäger, G. (2018). Are automatic methods for cognate detection good enough for phylogenetic reconstruction in historical linguistics? *CoRR*, *abs/1804.05416*. Retrieved from `http://arxiv.org/abs/1804.05416`

Rama, T., Wahle, J., Sofroniev, P., & Jäger, G. (2017). Fast and unsupervised methods for multilingual cognate clustering. *arXiv preprint arXiv:1702.04938*.

Ranacher, P., Neureiter, N., Gijn, R., Sonnenhauser, B., Escher, A., Weibel, R., … Bickel, B. (2021). Contact-tracing in cultural evolution: a bayesian mixture model to detect geographic areas of language contact.
doi: 10.1101/2021.03.31.437731

Richter, P. (1995). Estimating errors in least-squares fitting. *Telecommun. Data Acquisition Prog. Rep*, *42*(122), 107–137.

Rosenberg, N. A., & Nordborg, M. (2002). Genealogical trees, coalescent theory and the analysis of genetic polymorphisms. *Nature Reviews Genetics*, *3*(5), 380–390.

Rozas, J., Sánchez-DelBarrio, J. C., Messeguer, X., & Rozas, R. (2003). DnaSP, DNA polymorphism analyses by the coalescent and other methods. *Bioinformatics*, *19*(18), 2496-2497. Retrieved from `https://doi.org/10.1093/bioinformatics/btg359` doi: 10.1093/bioinformatics/btg359

Sand, A., Holt, M. K., Johansen, J., Fagerberg, R., Brodal, G. S., Mailund, T., & Pedersen, C. N. S. (2014). tqdist: A library for computing the quartet and triplet distances between binary or general trees. *BMC Bioinformatics*, *yy*(xx), ii-jj. doi: ???

Sokal, R. R., Oden, N. L., Legendre, P., Fortin, M.-J., Kim, J., Thomson, B. A., … Barbujani, G. (1990). Genetics and language in european populations. *The American Naturalist*, *135*(2), 157–175.

Swadesh, M. (1955). Towards greater accuracy in lexicostatistic dating. *International Journal of American Linguistics*, *21*(2), 121–137.

Thouzeau, V., Mennecier, P., Verdu, P., & Austerlitz, F. (2017). Genetic and linguistic histories in central asia inferred using approximate bayesian computations. *Proceedings of the Royal Society B: Biological Sciences*, *284*(1861), 20170706.

Verdu, P., Jewett, E. M., Pemberton, T. J., Rosenberg, N. A., & Baptista, M. (2017). Parallel trajectories of genetic and linguistic admixture in a genetically admixed creole population. *Current biology*, *27*(16), 2529–2535.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi: 10.1038/s41592-019-0686-2

Willems, M., Lord, E., Laforest, L., Labelle, G., Lapointe, F.-J., Di Sciullo, A. M., & Makarenkov, V. (2016). Using hybridization networks to retrace the evolution of indo-european languages. *BMC evolutionary biology*, *16*(1), 1–18.

Yang, M. A., & Fu, Q. (2018). Insights into modern human prehistory using ancient genomes. *Trends in Genetics*, *34*(3), 184–196.

Zlojutro, M., Tarskaia, L. A., Sorensen, M., Snodgrass, J. J., Leonard, W. R., & Crawford, M. H. (2009). Coalescent simulations of yakut mtdna variation suggest small founding population. *American Journal of Physical Anthropology: The Official Publication of the American Association of Physical Anthropologists*, *139*(4), 474–482.