Master Thesis

Submitted in Partial Fulfillment of the Requirements
for the Degree of
Master of Arts in Computational Linguistics

Mismatches between phylogenetic trees in Historical Linguistics

Author:
Marisa Delz

1st Supervisor:
Prof. Dr. Gerhard Jäger
2nd Supervisor:
Prof. Dr. Detmar Meurers

Seminar für Sprachwissenschaft
Eberhard-Karls-Universität Tübingen

April 2014

Ich versichere, dass ich die Arbeit selbständig und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe. Alle Stellen und Personen, welche mich bei der Vorbereitung und Anfertigung der Abhandlung unterstützten, wurden genannt und Ausführungen, die wörtlich oder sinngemäss übernommen wurden, sind als solche gekennzeichnet.

Tübingen, den April 17, 2014

Marisa Delz

Zusammenfassung

Die Phylogenie bietet Rechenverfahren, die für die (Computer-) Linguistik angepasst werden können. Einige dieser Methoden können aufgrund der Gemeinsamkeiten beider Bereiche in die historische Linguistik übernommen werden. Diese, für die Linguistik angepassten und modifizierten Methoden, können angewandt werden, um Geschichte und Entwicklung von Sprachen zu untersuchen, wobei diese Erkenntnisse zu neue Ansätze führen. Eine dieser Herangehensweisen ist der Vergleich zweier Bäume. In der Phylogenie werden Bäume hauptsächlich verglichen, um Rekonstruktionsmethoden zu testen.

Diese Arbeit fußt auf der Idee, durch den Vergleich der Bäume Unterschiede festzustellen. Um Abweichungen zwischen ihnen berechnen zu können, werden zwei Arten von Bäumen, Sprach- und Konzeptbäume, verglichen. Der Sprachbaum stellt die Geschichte der Sprachen dar, während der Konzeptbaum die evolutionäre Vergangenheit einer bestimmten Repräsentation eines Wortes zeigt. Konzept- und Sprachbaum werden mit phylogenetischen Methoden verglichen. Eines dieser Verfahren ist die Berechnung der Distanz zwischen Bäumen. Die zugrunde liegenden Daten für diese Bäume werden von der ASJP Datenbank bereitgestellt (Wichmann et al., 2012). Mit Hilfe dieser Daten sind linguistische Rekonstruktionsalgorithmen, wie der dERC Algroithmus (Jäger, 2013), in der Lage, sinnvolle Bäume zu konstruieren. Diese können dann automatisch verglichen werden. Die dadurch festgestellten Abweichungen können mit linguistischem Fachwissen interpretiert werden. Dies ermöglicht Einblicke in die Entstehungsgeschichte von Sprachen. Die Unterschiede der Bäume können dann in einem evolutionären Netzwerk visualisiert werden.

Abstract

The field of phylogenetics provides computational methods which can be adapted into (computational) linguistics. Due to parallels between the two fields, the interest of combining both arose. The adapted and modified methods can be used to study the history and evolution of languages and therefore new approaches emerged. One approach is the comparison of two trees. Up to now, trees were only compared to test different reconstruction methods.

This thesis exploits the idea of tree comparison for the detection of mismatches. To discover these mismatches, two types of linguistic trees are compared. These trees are so called language and concept trees. The language tree represents the history of languages, whilst concept trees display the evolutionary history of a representation of one specific word. The concept and language tree are compared using popular methods from phylogenetics. One of these methods is the computation of the distance between trees. The underlying data for these trees is provided by the ASJP database (Wichmann et al., 2012). Using this data, linguistic reconstruction algorithms such as the dERC (Jäger, 2013) are able to construct proper linguistic trees which can be compared automatically. The detected mismatches between the trees can be interpreted using linguistic background knowledge to get insights in the evolutionary history of languages. Within an evolutionary network, these mismatches can be depicted by reticulations.

Acknowledgements

I would like to express my greates gratitude to the people who have helped and supported me throughout my project.
I am truly and indebted grateful to my first supervisor Prof. Dr. Gerhard Jäger for his valuable guidance and support throughout my project and my theses. Without his hints during my research and writing phase it would have been more difficult to finish this project. His support and knowledge helped me to establish the approach. I would also like to thank him for providing and preparing the language data used in my approach. I want to thank my second supervisor Prof. Dr. Detmar Meurers as well. His hints and remarks on the structure of the thesis helped me a lot in making my thought more concrete. Additionally, I would like to thank Johann-Mattis List for explanations and insights on the construction of networks. Besides, I would like to thank Prof. Dr. Daniel Huson for his helpful suggestions on the phylogenetic program Dendroscope. A great thanks goes also to Johannes Wahle for insightful discussions, corrections, suggestions and moral support. Heike Cardoso for corrections and suggestions on my thesis.

Contents

List of Figures

List of Tables

1 Introduction

The use of computational methods in historical linguistics has become more and more popular in recent years and with it the usage of phylogenetic methods. In phylogenetics, evolutionary relationships between organisms are studied. These relationships can be represented in a phylogenetic tree. The idea of using a tree as metaphor goes back to Darwin, who introduced a tree of life. The tree of life describes the classification of organisms according to their evolution and relationship. This metaphor was adopted into linguistic. A pedigree for the Indo-European languages was introduced by August Schleicher. He was one of the first, who used the metaphor of a tree within linguistics. A pedigree expresses the evolution and relationship between languages. The adoption of the metaphor is not the only parallel between phylogenetics and linguistics. Darwin already noticed that there are parallels between biology and linguistics and Atkinson and Gray (2005) described this assumption in more detail. According to these parallels, the adaption of phylogenetic methods in linguistics increased.
New approaches are developed using the computational methods from phylogenetics, for example the detection and explanation of language evolution. The computational methods are used to automate processes like the reconstruction of trees. Linguistic trees indicate the classification of languages due to their evolutionary history.
Within this paper, there are two types of linguistic trees, a language tree and a concept tree. The ASJP database (Wichmann et al., 2012) is used to reconstruct both. The database includes several hundred languages and 40 concept representations for each one. The 40 concepts can be seen as a list of words covering the basic vocabulary. A language tree represents the history of languages, whereas a concept tree describes the history of a concept. The trees can be compared to detect a mismatch. This can either be done by using already existing algorithms from phylogenetics or by modifying an approach. Within the new approach, distance measurements are used to compare the trees. The mismatch can be indicated by reticulations which can be interpreted as evolutionary events. The method is implemented and provided by the author.

In this paper, a new approach to detect evolutionary events using distances is introduced. The second chapter is an introduction to historical linguistics, phylogenetics and the parallels between the two fields. The next chapter provides background information on the data and algorithms used for the implementation. The reconstruction of the linguistic trees is explained and an improved version is described. In the fourth chapter, phylogenetic methods to detect evolution-

ary events are described. The different methods are compared and the network constructed with the best algorithm is evaluated. The evaluations provides advantages and disadvantages for the use of the algorithm on linguistic data. The last chapter states the idea of the new approach. Different distance measurements are described and compared. The best one is used for further analysis of the evolutionary events. The result of the program is a network which is able to visualize the reticulations.

2 Historical Linguistics and Phylogenetics

Within linguistics, historical linguistics is the study of language history and language change. Over the years, language is passed on from one generation to the other. The children learn the same language as their parents and so on. With this transmission of language, changes can appear from generation to generation. Language is a system which develops over time. The change within a language are studied in historical linguistics. If the language changes, the words within a language might also undergo change. This change can be independent of the change within the corresponding language. The history of words is determined by language history. Within historical linguistics, both the history of the language and of its words are studied. During the process of language evolution, evolutionary events happen. An example of this is the split of an ancestor language forming two languages. These events are responsible for the development of a language and for the present day languages. Because language is a changing system, there might be no stable version of any language.

Evolution is a process which is not only present within linguistics, but also within other scientific fields. In Biology, evolution is the transmission of genes due to reproduction of individuals within a population. The evolution of for example animals, plants or bacteria is better explored than the evolution of language. For several years, computational methods have been used in biology and a new subfield emerges, namely bioinformatics.

Phylogenetics is a field within bioinformatics which studies and analyses the evolutionary relationship between groups. Within biology, phylogenetics provide methods for modelling evolutionary relationships between organisms. These methods are very helpful for the research of biological evolution. Evolutionary relationships can be displayed within a phylogenetic tree, which can be reconstructed automatically. Evolutionary events can be displayed within a phylogenetic network, which can also be reconstructed automatically. The implemented methods provide a fast way for the reconstruction of evolution.

Within linguistics, evolution and language history is explored by field workers. They collect information about languages manually and reconstruct evolutionary relationships by hand. Parallels can be drawn between biological and linguistic evolution. These parallels are the motivation to provide an automatic way to reconstruct and model the evolutionary relationships between languages. The phylogenetic methods can be adapted to linguistic research. Phylogenetic trees are used to reconstruct evolutionary relationships between languages and phylogenetic networks represent evolutionary events within linguistics.

2.1 Historical Linguistics

Language is one of the fastest changing systems in humanity. Scientists in the field of historical linguistics analyse the change within languages. Language change can affect different areas of a language, for example pronunciation, orthography and grammar. Depending on the type of change, different areas tend to be more affected than others. Nevertheless, it takes some time until the change is embedded in a language. Therefore, historical linguistics studies language change during a time period and automatically include the history of the languages.
Historical linguistics is sometimes called "*diachronic* linguistics" (Campbell, 2013, p. 3). The word *diachronic* comes from "Greek *dia-* through + *chronos* time + *-ic*" (Campbell, 2013, p. 3). Diachronic linguistics deals with the language and its temporal change during time.

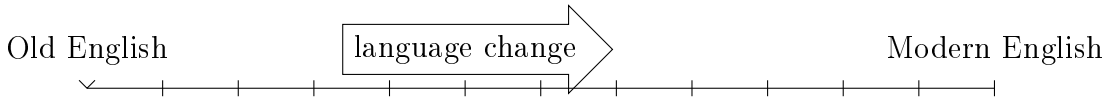


Figure 2.1: Diachronic Linguistics

Its counterpart, *synchronic* linguistics, deals with the language at a specific point of time (Campbell, 2013). Most likely with a particular phenomenon within a language. This could be the grammar of a language like Old English or new established words within a language in a stipulated year.

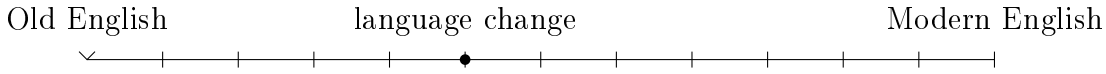
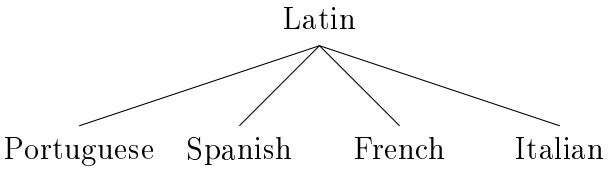


Figure 2.2: Synchronic Linguistics

As said above, language change can affect different areas of a language. The question arises which type of change is responsible for the language change and which area is affected. There are two different types of language change. First, the change within a language and second the change occurring between unrelated languages.
The subfield, which deals with the change within a language, is called *philology* (Campbell, 2013). The change is due to *internal factors*. Internal factors can be sound change, syntactic change or morphological change and therefore affect different areas, for example pronunciation or orthography. (A. M. S. McMahon, 1995). These changes might be due to dialects affecting a language or because people pronounce words differently. Due to the transmission of language from one generation to another, the change is embedded into the language. Transmission is also a reason to pass vocabulary and generalisations. Generalisations simplify the

learning of different word forms without learning each single word and its form (Nowak & Krakauer, 1999; Jackendoff, 1999). An example is the verb system in English and German. Verbs can be either strong or weak. Strong verbs have different inflection forms, whereas the inflection of weak verbs is always the same. Therefore, strong verbs and their forms need to be learned one by one, whereas weak verbs can be inflected due to a generalisation rule (Delz, Layer, Schulz, & Wahle, 2012). People start using the generalisation rule also for some strong verbs, therefore strong verbs change into weak verbs. There are other factors, like frequency of the verbs, which play a role during this process. The lower the frequency, the higher the probability that a strong verb changes into a weak. A high frequency of a strong verb indicates a low probability that the verb changes into a weak. Nevertheless, this is an example for an internal factor of language change. The subfield, which studies the change between related languages, is called *comparative linguistics* (Campbell, 2013). This change is revealed while comparing (related) languages (Campbell, 2013). The change is due to *external factors*. External factors can also be sound change, syntactic change or morphological change. While comparing related languages, the change can be caused by a common ancestor. In this case, two languages are descendants of a common ancestor. At some point in time, the ancestor language splits into two languages. These two languages develop and change separately. By comparing them, similarities can be detected and the change can lead back to the common ancestor language. The example in (1), illustrates this phenomenon. Latin is the common ancestor of Portuguese, Spanish, French and Italian. The four descendants of Latin split up at some point in time. Due to sound change, syntactic change and morphological change new languages emerged and can be distinguished from their ancestor and from the other related languages. Nevertheless, a connection can still be drawn between them and their common ancestor.

(1)



On the other hand, unrelated languages can also show parallels. These are due to language contact. During language contact, words can be adapted into a language and the language assimilates the word according to its own phonology and syntax. Language contact can also lead to the emergence of new languages. A *pidgin* language arises due to the contact between different people who do not share a common language (Campbell, 2013). The people need a language which enables them to successfully communicate. A *creole* language occurs from a pidgin

language (Campbell, 2013). The pidgin language becomes a native language spoken by people in an area and due to this development, the language becomes a creole language. Therefore, a creole language is a new emerged language of a population emerged from language contact. The comparison of languages, either related languages or unrelated languages, can be used for detecting language change. Historical linguistics can evince language contact and common ancestors. The analysis of language change and history leads automatically to the analysis of word history and change. This subfield is called *etymology* (Campbell, 2013). Words can have their own history independent from the one of the language. There are parallels, for example languages can have a common ancestor so do words and there is language contact so words can be borrowed. If two languages are related, some words in these languages can be *cognates*. Cognates are related words which are descendants of a common ancestor. This can be seen in example (2) taken from (List, 2013).

(2) a. Latin: computare (meaning: to count) $\Longrightarrow$ Spanish: contar (meaning: to count)
b. Latin: computare (meaning: to count) $\Longrightarrow$ French: compter (meaning: to count)
c. Latin: computare (meaning: to count) $\Longrightarrow$ Italian: contare (meaning: to count)

All languages are descendants of the Latin language and the words are cognates, which means they are all related to the ancestor word *computare*. Words can also be cognates, if a language is not their common ancestor. This can be seen in the next example. English derives from the Germanic language, but has a similar word for *count* as the Romance languages.

(3) a. Latin: computare (meaning: to count) $\Longrightarrow$ Old French: conter (meaning: to count) $\Longrightarrow$ English: to count

This is the case, because *count* is borrowed from Old French (List, 2013). Borrowing is due to language contact, where one language borrows a word and its meaning from another language. Borrowed words are also called *loanwords*. In example (4), the word *discus* is borrowed from Latin into the Middle High German language (spoken around 1050 and 1350) and into Old English. The words are still cognates, but their meaning changed during time.

(4) a. Latin: discus (meaning: disc, a circular plate) $\Longrightarrow$ German: Tisch (table, a plate with legs)

b. Latin: discus (meaning: disc, a circular plate) $\Longrightarrow$ English: Dish (is still a plate)

Borrowing is a change within two languages due to language contact. Language contact can only be the case between two language at the same point in time. The languages come into contact because speakers of one language meet speakers of another language. Words are borrowed into a language because of different reasons. One of the main reasons is that the borrowing language does not have a word for describing a specific meaning or a concept. This specific meaning can be for example religious terms or the name of a product. The borrowing happens under specific circumstances. Bilingual speakers might use words from one language in the other language, because the second language cannot describe the meaning. This can be people living near the border or having parents which have different native languages. Another example is travel. People who are travelling or moving to other countries might pick up words and embed them into their own language. Thanks to ships, locomotives, cars and other vehicles travelling and trading is a lot easier. Products are imported into countries and it is likely that the description will be imported, too. The words are then assimilated by the language. People adapt loanwords while adapting the phonology and/or orthography of the word. After some time, loanwords might not be recognized as such because they are fully integrated into the language. Borrowing words is a historical process. The borrowing itself is short but the adaption into the language is a long historical process and the loanword is the result of it (Haugen, 1950).

Borrowing takes place between two existing languages. Therefore, the languages already have a so called basic vocabulary. Swadesh (1955) made a list of words which, according to him, belong to the basic vocabulary of a language. The words are non-cultural and universal and are present languages. The first list contained 100 words and most of the words are cultural concepts like numerals or animals (Swadesh, 1955). According to Swadesh (1955), these words and concepts are resistant to language evolution and therefore also resistant against borrowing. Nowadays, it is said that the words contained in the Swadesh list are less likely to be borrowed. This assumption does not exclude that some words from the basic vocabulary might be borrowed. This is the case for the English word *mountain*.

(5) a. Old French: montaigne $\Longrightarrow$ English: mountain

The word is borrowed from Old French, although it belongs to the basic vocabulary (Joseph & Janda, 2003). this example points out that words from the basic vocabulary are not resistant against borrowing, but are less likely to be borrowed.

Mountain is one of the few words which are loan into other languages.
Swadesh (1955) compiled his list manually, just using his intuition. The list includes words represented in most of the languages and most of them are less likely to be borrowed. Therefore, the list can be used as a representation of basic vocabulary.
Nevertheless, it was shown that not all words in the Swadesh list are resistant against borrowing. This is a good example to show, that each word in a language has its own individual history. Etymology is not the main purpose within historical linguistics. It goes hand in hand with philology and comparative linguistics. Languages and words can have a common ancestor, although this ancestor might not be the same. Language contact leads to borrowing of words from one language into the other. All of this is studied and analysed in the field of historical linguistics. Etymology can be seen as a by-product of historical linguistics (Campbell, 2013). Nevertheless, it is important. Words can have different histories than the language they belong to. If we understand the history of the words within a language, the reconstruction of the language history might be easier. The main goal of historical linguistics is to understand language change in general (Campbell, 2013). Etymology, philology and comparative linguistics are all needed for a full understanding of language change.
In the past, the studies and analysis focused on "how" languages change and the theory behind the language change (Campbell, 2013). The question "why" languages change was addressed in present studies. Campbell (2013, p. 5) found the right words to explain the studies within the fields of historical linguistics:

> "Today, we can say that historical linguistics is dedicated to the study of "how" and "why" languages change, both to the methods of investigating linguistic change and to the theories designed to explain these changes"

The methods to investigate linguistic change and for reconstructing language history are taken from another scientific field, namely *Phylogenetics*.

2.2 Phylogenetics

In phylogenetics the evolutionary relationship between groups is studied and analysed. In biology, evolution is a process where individuals within a population pass their genes to the next generation due to reproduction. Phylogentics analyses the evolutionary relationships between individuals, populations and organisms.
The basic idea goes back to Darwin. He constructed a *tree of life* which represents the classification of organisms and their relationship. The genealogical

development of the organisms is known as *phylogeny*. The first sketch of a tree from Darwin is shown in figure 2.3 (Eldredge, 2005).

Figure 2.3: Darwin's first sketch of a tree

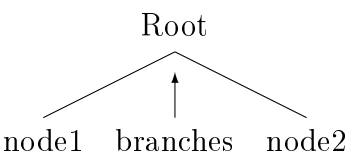
The idea of using a tree to represent relationships is way older. It is actually a bible phrase (Penny, 2011). Within families, pedigrees or family trees are well known to represent the relationships between the persons in a family. So why not use this idea for representing related individuals within a population or between organisms? And why not use the metaphor of a tree for representing the relationships between languages? Darwin uses the idea for representing related organisms and their genealogical development. He didn't know anything about genes and his idea was triggered by observations (Eldredge, 2005). Darwin only knew that "[o]rganisms resemble their parents, [...] the variation in the appearance of organisms within a single species is heritable, [...] [and that] more organisms are produced each generation than can possibly all survive and themselves reproduce" (Eldredge, 2005, p. 69).
Out of this thought, the theory of *natural selection* emerges. Lecointre (2006, p. 13) stated the idea of it in a clear way:

> "Under the particular environmental conditions of a given moment in time, certain variants are favored and become more numerous because they leave behind more descendants than do competing variants. "

A population emerges out from reproduction and survives in terms of natural selection. Darwin's idea was to study the inherited characters and genes. Inherited genes are transferred from one generation to the other and related species can be detected. Therefore, genes are used to detect relationships between organisms and with explicit methods the ancestral species or even the history of a species

can be reconstructed.
Willi Hennig uses Darwin's idea to introduce his approach on *phylogenetic systematics* in his German work *Grundzüge einer Theorie der Phylogenetischen Systematic* and in his English work *Phylogenetic systematics*. Phylogenetic systematics is nowadays called *cladistics*. In cladistics, evolutionary relationships among species are recognized and grouped together according to their common ancestor (Lecointre, 2006). Such a group of organisms is also called *taxon* and normally it is associated with a scientific name. If the group is not associate with a name, a new name is created to describe this group. The theory and practice of naming and grouping organisms is called *Taxonomy* (Wiley & Lieberman, 2011). The diversity of the organisms in a group is relevant for the evolution of the organism. Each organism has a set of characters and the state of a character is used for discriminating it within a group of organisms. It is assumed for each character to have *homologous* states. Homologous meaning similar, where the states can be identical or differ slightly. Not all character states are homologous but certain resemblances might be convergent (Lecointre, 2006). A data matrix is used to code the characters, because not homologous characters cannot be detected immediately (Lecointre, 2006). With the help of the data matrix, all possible evolutionary trees are build. The trees integrate the smallest number of evolutionary events needed by the data matrix for building the tree. "We keep only the most parsimonious tree - the one with the fewest number of evolutionary steps" (Lecointre, 2006, p.16-17).
The illustration of trees which are reconstructed by using different methods can differ. Before I introduce the two main illustration of the tree which are used within phylogenetics, the first created pedigrees in figure 2.4 is shown. This pedigree was created by Haeckel (1874), who refines Darwin's idea of an evolutionary tree. Such a tree is one possibility to illustrate a pedigree. Trees can be used for representing all kind of relationships in a clear and intuitive way. Therefore, trees became a famous instrument to represent relatedness among different organisms. As can be seen in figure 2.4, a tree consists out of a root, branches and nodes. The root which represents the ancestor can either be at the bottom or at the top of the tree. From the root, different branches emerge which lead to a node.

(6)



In (6) the root is at the top. Therefore, the tree is called *top-down*. In contrast to that, the pedigree of Haeckel (1874) is *bottom-up*, due to the fact that the root

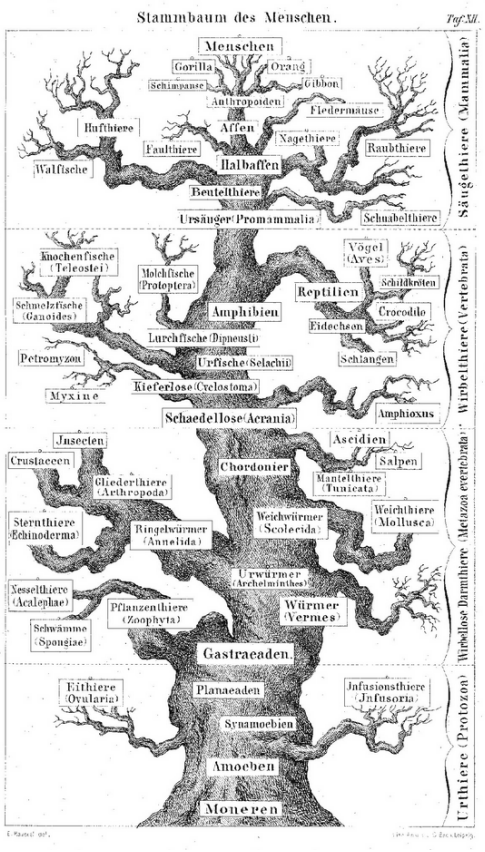


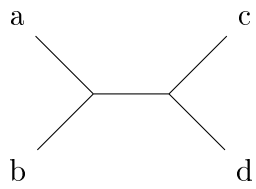
Figure 2.4: Haeckel's famous pedigree of man

is at the bottom. The root can also appear on the right or on the left side.
Lecointre (2006, p.21) states a mathematical definition of trees:

Definition 2.1 *A tree is a noncyclic, connected graph.*

The tree has to be noncyclic, all nodes are connected with their ancestor and all branches are at least binary branched. Binary branching refers to the fact that each node has two branches each pointing to one child. The fact that two nodes are only linked by one branch is called *noncyclic*. Such a graph is the basis of phylogenetic trees.
There are two different sorts of phylogenetic trees, namely *unrooted* and *rooted* trees. The examples below are taken from Lecointre (2006, p.22) where for simplicity reasons only four taxa are used.
The advantage of unrooted trees is that "they are consistent with a limited number of rooted trees" (Wiley & Lieberman, 2011, p.101). The illustration in (7) represents one out of four unrooted trees which consists of four taxa.
The illustration of an unrooted tree is stable, but the labelling of the nodes change. Through this change, different unrooted trees emerge.

(7)



Huson, Rupp, and Scornavacca (2010, p.25) state a formal definition of an unrooted tree:

Definition 2.2 *Given a set of taxa χ, a phylogenetic tree T on χ consists of a tree $T = (V, E)$, in which all nodes have degree $\neq 2$, together with a taxon labeling λ: $\chi \to V$ that assigns exactly one taxon to every leaf and non to any internal node.*

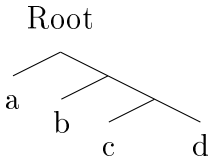
In the definition, V indicates the set of nodes, E indicates the set of edges or branches and the phylogenetic tree indicates an unrooted tree. The number of edges connected to a node is called degree of a node. Indegree indicates the number of incoming edges, whereas outdegree the number of outgoing edges. The set of taxa in (7) would be $\chi = \{a, b, c, d\}$ and the tree would be the same graph without nodes shown in example (7). Each label in the set would be assigned to one node by chance. Therefore, the illustration of the tree is stable, but not the labelling of the nodes.
According to the mathematical definition in 2.1, the trees need to be a noncyclic connected graph. Noncyclic in the sense of Lecointre (2006) means two nodes are linked by only one path. Both inner nodes (which are not labeled in 2.3) are connected with each other and additionally they are also connected with their children. This is the reason, why the unrooted tree is noncyclic but not binary branching, because the inner nodes are connected with three nodes and not with two. Therefore, the unrooted tree in (6) might not be binary branching but it fulfils the definition.
An unrooted tree can be transformed into one or more rooted trees, whereas each node in the set can be the root. Huson, Rupp, and Scornavacca (2010) stated a formal definition of rooted trees:

Definition 2.3 *Given a set of taxa χ, a rooted phylogenetic tree consists of a rooted tree $T = (V, E, \rho)$ and the taxon labeling $\lambda : \chi \to V$ that assigns exactly one taxon to every leaf and none to an internal node. All nodes, except ρ, must have degree $\neq 2$.*

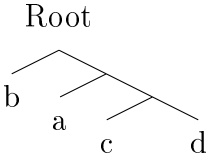
The taxa set $\chi = \{a, b, c, d\}$ includes all taxa and they are assigned to the nodes of the raw tree. Diverse trees can be created by varying the taxon which is used

as root node. Below all possible rooted trees are listed which result from the one unrooted tree represented in (6).

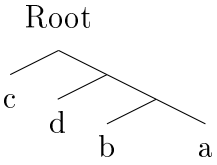
(8) a. The tree is rooted on a:



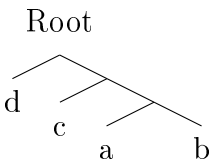
b. The tree is rooted on b:



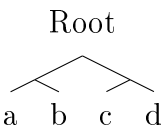
c. The tree is rooted on c:



d. The tree is rooted on d:

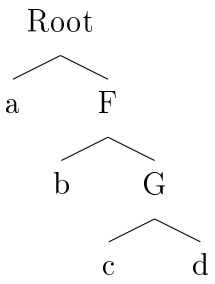


e. The tree has a midpoint root



Hennig's idea was that the tree should be rooted on an outgroup (Lecointre, 2006). Depending on the outgroup, dissimilar trees can be built. All rooted trees should confirm the definition in 2.1, therefore the trees need to be connected noncyclic graphs. In all rooted trees the nodes are connected and therefore noncyclic.

(9) a. The tree is rooted on a:



The representation in (9) gives us a clear picture of a binary branching tree. Each node, namely the root and the inner nodes B and C, are connected with two children. Therefore, the tree establishes a classical example of a binary tree. An alternative to phylogenetic trees are phylogenetic *networks* (Huson, Rupp, & Scornavacca, 2010). Networks and trees do not differ much from each other and in a broad sense, a network can be seen as a cyclic tree. Depending on what should be represented, either trees or networks should be chosen. As stated above, trees

are used for representing the relationship between organisms or genes and their evolutionary history. Networks can be used to represent evolutionary events of an organism or a gene.
A general definition of a network is given by Huson, Rupp, and Scornavacca (2010):

Definition 2.4 *A phylogenetic network is any graph used to represent evolutionary relationships (either abstractly or explicitly) between a set of taxa that labels some of its nodes (usually the leaves)*

Explicit networks represent evolutionary events, especially reticular events like horizontal gene transfer, where a gene is transferred between two unrelated organisms. *Abstract networks* are used to visualize (incompatible) taxasets (Huson, Rupp, & Scornavacca, 2010).
Similar to trees, networks can also be divided into two groups, namely *unrooted* networks and *rooted* networks. Both are defined analogously to unrooted and rooted trees.
Unrooted networks can be compared to an unrooted tree: there is no root and the edges can be spread to all sides. A definition of an unrooted network is given by Huson, Rupp, and Scornavacca (2010):

Definition 2.5 *An unrooted phylogenetic network N on χ is any unrooted graph whose leaves are bijectively labeled by the taxa in χ.*

Rooted networks on the other hand are comparable to rooted trees. Their branches emerge from one root and are built up to a tree-like network. A definition of a rooted network is given by Huson, Rupp, and Scornavacca (2010):

Definition 2.6 *A rooted phylogenetic network N on χ is a rooted DAG [(direct acyclic graph)] whose set of leaves is bijective labeled by the taxa in χ. Any node of indegree ≥ 2 is called reticulate node and all others are called tree nodes. Any edge leading to a reticulate node is call[ed] a reticulate edge and all others are called tree edges.*

Consequentially, unrooted and rooted networks are similar to unrooted and rooted trees. The relevant difference is that networks include the representation of evolutionary events whilst trees don't.
If we want to link the terms of unrooted and rooted networks to explicit and abstract networks, the unrooted could be linked to the abstract while the rooted establishes a link to either the abstract or the explicit networks.
This is due to the fact that an unrooted network can only be abstract. It does not explicitly represent all evolutionary events, but it is able to represent incompatible

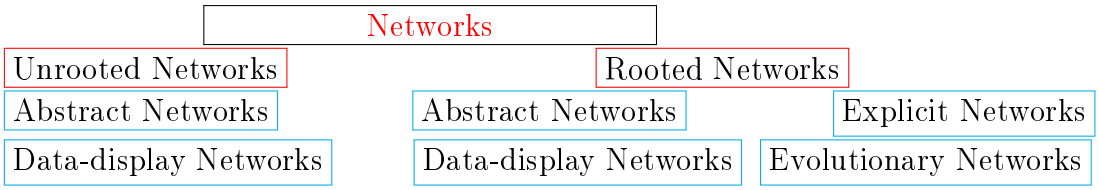


Figure 2.5: Summary of the different networks

taxasets. Whereas the rooted can be both, abstract or explicit depending on the evolutionary events it represents. In other words, if a network (unrooted or rooted) represents these evolutionary events it is explicit, otherwise it is abstract (Huson, Rupp, & Scornavacca, 2010).
Further, networks can be divided into *data-display* and *evolutionary* networks (Morrison, 2011). Here the connection between unrooted and rooted networks can also be drawn. Data-display networks are abstract and either rooted or unrooted. Even if the taxa sets are incompatible the data-display network can indicate the relationship between the samples. In this case, the network functions as a diagram representing possible relationships without making an assumption on evolutionary change. Evolutionary networks are rooted and explicit. The evolutionary change and history can be indicated and visualized. The root depicts the ancestor, the branches lead the descendants, and along that path, evolutionary change takes place (Morrison, 2011).
To avoid confusion, an overview of all representations of networks is displayed in figure 2.5:

2.3 Parallels between Biology and Linguistics

Evolutionary relationships are studied and analysed in different scientific fields. As mentioned above, biological evolution implies reproduction between individuals of a population and the transmission of genes to the next generation. Phylogenetics provide computational methods to reconstruct evolutionary relationships, history and events. The field of phylogenetics is a subfield of bioinformatics. Therefore, the computational methods which are developed and implemented in phylogenetics are based on biological data and are used to analyse the above. The functionality of the methods can be extended to other scientific fields, as in our case linguistics.
In linguistics, language evolution studies the origin and development of languages and its words. It assumes the transmission from one generation to the next which can be compared to the transmission of genes via reproduction. The “children” inherit the genes or words. Due to this inheritance, the children have the same ancestor or parents which define their relation. Languages and words can as well

be related and share a common history. Therefore, as these same features are present within linguistics and biology, phylogenetic methods can be used to detect these between languages, too.
These parallels have already been recognized by Darwin (1871, p. 57-58):

> "The formation of different languages and of distinct species, and the proofs that both have been developed through a gradual process, are curiously the same. But we can trace the origin of many words further back than in the case of species, for we can perceive that they have arisen from the imitation of various sounds, as in alliterative poetry. We find in distinct languages striking homologies due to the community of descent, and analogies due to a similar process of formation."

It is not surprising that Darwin's idea of the tree of life can also be found in the field of linguistics. The following pedigrees in figure 2.6 by Haeckel (1874) and 2.7 by Schleicher (1873) show the first pedigrees of the Indo-European language family.

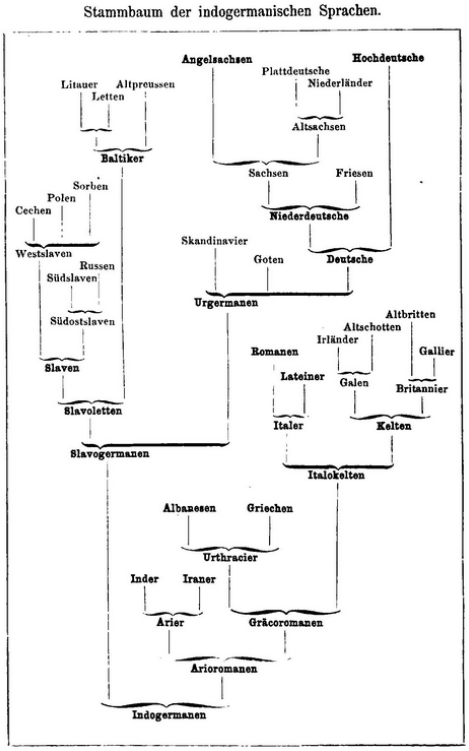


Figure 2.6: Haeckel's pedigree of the Indo-European languages

Ernst Haeckel was a biologist, who introduced Darwin's theory of the tree of life to his friend and linguist August Schleicher (Atkinson & Gray, 2005). Both used the metaphor of trees independently to depict language history. The similarities between the two approaches are revealed through Darwin's theory. In their corresponding scientific fields, both authors are famous for transferring language

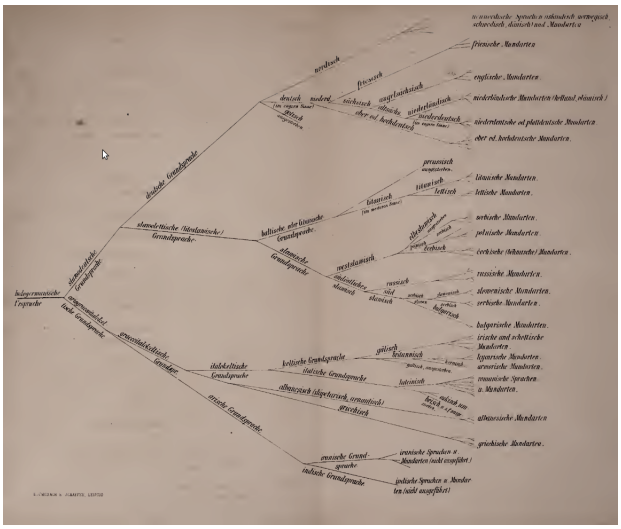


Figure 2.7: Schleicher's pedigree of the Indo-European languages

history to a tree model. Thanks to their friendship and letter exchange, initiated the ongoing contact between biologists and linguists. On the surface, they use the same method for representing evolution and its relationships. However their theories are distinct due to their background knowledge.

The question is what are the evolutionary events which might take place during the course of time? Neither of the above trees reveal these events. According to Atkinson and Gray (2005) concepts in biological and linguistic evolution can be matched. Those correlations are summarized in table 2.1, taken from Atkinson and Gray (2005, p. 514).

Biological evolution	Linguistic evolution
Discrete characters	Lexicon, syntax, and phonology
Homologies (Orthology, Paralogy)	Cognates
Mutation	Innovation
Drift	Drift
Natural selection	social selection
Cladogenesis	Lineage splits
Horizontal gene transfer (Xenology)	Borrowing
Play hybrids	Language Creoles
Geographic clines	Dialects/dialect chains
Fossils	Ancient texts
Extinction	Language death

Table 2.1: Conceptual parallels between biological and linguistic evolution

A detailed explanation of every parallel between biological and linguistic evolution is not vital to our purpose. The significant fact is that such parallels exist. They result from the evolution process. These unalterable events are what we are interested in. If phylogenetic methods help to reconstruct different aspects

of biological evolution, the same methods can assimilated by linguistics to reconstruct language evolution, relationships between languages, language history and evolutionary events. If we can adapt the reconstruction methods, we can also adapt other phylogenetic methods like algorithms for tree comparison.

3 Reconstructing Trees with Linguistic Data

Computational methods found their way into linguistics. Within the field of computational linguistics several methods have been studied and adapted from other fields. Therefore, it is not surprising that those of phylogenetics are adapted to linguistics. Especially, since linguistics and pyhlogenetics show the parallels explained in the preceding chapter. Nevertheless, they cannot be integrated "one-to-one". A useful approach is the reconstruction of phylogenetic trees. Within evolutionary linguistics the automatic reconstruction is profitable and a counterpart to the trees created manually by experts of the specific language. To enable the reconstruction, a linguistic database and a reconstruction algorithm is needed.

In this project, the linguistic database used is the **A**utomated **S**imilarity **J**udgement **P**rogram database (Wichmann et al., 2012). This database provides phonological representations for a fixed set of words within different languages. There are hundreds of languages from many different language families contained in the database. Therefore, it serves as a sufficient huge database which can be used to reconstruct linguistic trees (Jäger, 2013; Brown, Holman, Wichmann, & Velupillai, 2008). The phonological representations of the words can be compared by an algorithm to measure their similarity. A reconstruction algorithm needs this information to compute a tree.

There are several reconstruction algorithms within the field of phylogenetics. Of course, we want to use the fastest and most efficient algorithm. According to Huson, Rupp, and Scornavacca (2010), FastME (Desper & Gascuel, 2002) is the most recent algorithm among the distance-based. The algorithm contains several methods to compute a phylogenetic tree and one of these methods is the best for our purpose as it is explained in the following section. The algorithm can be almost completely incorporated to compute a linguistic tree. However, it is only successful in doing so, if enough data is available. The needed adaption to use the algorithm within linguistics does not consists of changing it but creating methods to produce the correct input. If this is achieved, the algorithm can compute linguistic trees.

Two sorts of trees can be computed, namely *language trees* and *concept trees*. The idea behind this classification is stated in section 3.3.1.

3.1 ASJP Database

The database of the **A**utomated **S**imilarity **J**udgement **P**rogram (ASJP) is designed for automated classification of the world's languages through lexical com-

parison (Wichmann et al., 2012). It can be used to compare pairs of words and to verify whether the words are similar. Depending on this judgement, the similarity between languages can be stated and the languages classified.
For a lexical comparison between words, a list of words and their corresponding translation into several languages is necessary. Wichmann et al. (2012) started with the already known list of Swadesh (1955). This list contains 100 words, which are believed to be almost stable within languages and are less likely to be borrowed. These 100 words are assumed to form the basic vocabulary and are therefore present in most of the world's languages. The 100 word Swadesh list can be found in appendix A. The ASJP Database includes 245 languages into which the 100 words of the swadesh list are all translated manually.
ASJP uses its own orthography. All words are represented with respect to it. The orthography consists of 41 symbols, 7 vowels and 34 consonants, and can also be found in appendix B (Brown et al., 2008). It is similar to the International Phonetic Alphabet (IPA) (Association, 1999). The IPA is a collection of signs which describe a sound of the human language. Each sign represents one sound. These signs provide an relatively unbiased standard to encode the pronunciation of a word. The ASJP orthography is a simplified version of the IPA. Each ASJP symbol can represent one sound, but it may also be the case that it represents a combination of IPA sounds or covers a range of these (Brown et al., 2008). Nevertheless, the most common sounds within the world's languages can be represented with the symbols and rare sounds are represented by the closest sound and its corresponding symbol (Brown et al., 2008). Therefore, a phonetic representation of a word is provided. This facilitates the comparison between words and the calculation of the similarity is no longer influenced by the orthographic aspects of each language.
In a first study, Brown et al. (2008) calculated the similarity of languages in three steps:

1. Calculation of the LSP
2. Calculation of the PSP
3. Calculation of the SSP

The **L**exical **S**imilarity **P**ercentage described by Brown et al. (2008, p. 6):

> "LSP is the number of items on the 100-item list for which two compared languages have words that are judged phonologically similar by ASJP, divided by the number of meanings on the list for which both of the languages have words, the result multiplied by 100"

The LSP can be represented formally:

$$\text{LSP} = \frac{\text{words phonologically judged on the 100-item list}}{\text{words with same meaning in both languages}} * 100$$

For each language pair, a percentage of their similarity is calculated. This percentage is stored in a list which represents a kind of database for the reconstruction of language trees. Within the trees, the similarity of the languages is graphically visualized (Brown et al., 2008). As stated in Brown et al. (2008), the genetic relationships between languages and not factors like language contact are reflected. This assumption cannot be confirmed with certainty. The length of words and the phonology of languages can have an impact on the results. Therefore, Brown et al. (2008) calculate the similarity between the phonological representations.
The **P**honological **S**imilarity **P**ercentage is described by Brown et al. (2008, p. 6) in the following way:

> " [PSP] is defined as the average similarity (calculated as for LSP) among pairs of words that do not refer to the same concept on the 100-item list."

The PSP can be represented formally:

$$\text{PSP} = \frac{\text{words phonologically judged on the 100-item list}}{\text{words without same meaning in both languages}} * 100$$

The PSP is calculated to compensate the problems and effects of long words and phonological similarity which arise during the calculation of the LSP. Words which differ semantically within the 100-words list are compared and the value is represented by the PSP.
The **S**ubstracted **S**imilarity **P**ercentage is the result of:

$$\text{SSP} = \text{LSP} - \text{PSP}$$

The SSP indicates a more precise value to calculate the similarity between two languages. Brown et al. (2008) claim that this value leads to more precise trees which are closer to the manually constructed language trees. Therefore, SSP is preferred over LSP. All SSP values are represented within a matrix. This matrix can be used to reconstruct language trees in phylogenetic programs (Brown et al., 2008).
One result of the study was the reduction of the 100 word list to a 40 word list. This is due to the fact that a 40 word list is easier to handle and the results of the similarity calculations are just as good as with the larger word list. The current database provides this 40-word list which can be found in appendix C. These are

the most stable among the world's languages (Wichmann et al., 2012). Besides the deletion of the 60 words and the addition of further languages, no further changes were made. The database consists of one file, containing the following parts:

- Specification of the ASJP version
- Internal information
- The 40-word list
- The ASJP symbols of the orthography
- A word list for each language

Each word of the 40 is numbered and represented in a single line, illustrated in (10):

(10) Number (tab) word
1 (tab) I
2 (tab) you
3 (tab) we

Each symbol of the orthography is also represented in one line. The word list for each language is more complex. Each list is stored with some additional information, illustrated in (11) and (12):

(11) LANGUAGE_NAME{classification used in WALS|classification used in Ethnologue@classification used by (Hammarström, 2010)}
properties of the languages, like number of speakers
numberOfConcept(WALS) word wordInASJPOrthography //

(12) STANDARD_GERMAN{IE.GERMANIC|Indo-European, Germanic, West, HighGerman, German, MiddleGerman, EastMiddleGerman@Indo-European, Germanic, WestGermanic, Franconian, HighFranconian}
1 52.00 10.00 90294110 ger deu
1 I iX //
2 you du //
3 we vir //
11 one ains //

The word lists of the languages might be the most important information for people working with the data.
In the following, the term *word* is replaced by the term *concept*. A word is the representation of a concept within a language. A concept is more than a simple

translation of the word, it represents a specific meaning. The translation of a word can only take place, if the other language does have a word which represents the same meaning. Other languages might not have a specific word, but can express the meaning of the word in a different way. An example is given below in (13).

(13) English: you
Chinese: neih deih (orthography of the ASJP database)

The English word *you* is one word addressing another person. In Chinese, this meaning is expressed with two words. Therefore, *you* cannot be translated word by word into Chinese, rather the concept of the meaning is translated using the two word representation of Chinese.
The information provided in this database can be used for reconstructing phylogenetic trees. It is the linguistic background information which can be used as input data for phylogenetic methods. We are looking for phonetic representations of words and those are provided within this database. Phonological representations can be seen as a uniform spelling format. Other databases only include lexical representations of the words or additional information which is not need for our purpose. The ASJP database includes all information needed to serve as input data for distance-based phylogenetic reconstruction methods.

3.2 The FastME Reconstruction Algorithm

Phylogenetics show the relationships and the shared history between different organisms. The unrooted and rooted trees, introduced in section 2 are used to illustrated the relationships and history of them. In computational biology, *phylogenetic inference* is responsible to compute an evolutionary tree which should be the most optimal tree to represent the history of the organisms. Phylogenetic inference is analogous to the induction of a family tree of languages (Jäger, 2013). There are basically two main groups to reconstruct trees, namely *sequence-based methods* and *distance-based methods* (Huson, Rupp, & Scornavacca, 2010). Sequence-based methods are also called *character-based methods*, whereas the first term is mostly used in computational biology and the second term is mostly used within linguistics. Therefore, I will use the term character-based methods instead of sequence-based methods.
The underlying process of both group of methods is *sequence alignment*. A sequence is a string containing the elements represented in the data set. These elements can be genes or words. Sequence alignment is the comparison between two or more sequences, whereas the comparison between two sequences is called *pairwise sequence alignment* and the comparison between more sequences is called

multiple sequence alignment (Huson, Rupp, & Scornavacca, 2010).
The character-based methods "search for a phylogenetic tree T that optimally explains a given multiple sequence alignment M" (Huson, Rupp, & Scornavacca, 2010, p. 33). The sequence alignment is the input for all character-based methods and due to the alignment and a specific method, a tree can be reconstructed. With the methods the optimal tree can be reconstructed according to a specific criterion like maximum parsimony or maximum likelihood.
According to Huson, Rupp, and Scornavacca (2010, p. 33), "distance-based methods usually construct a phylogenetic tree T from a given distance matrix D". The sequence alignment is the input for different methods which compute the distance matrix. The *hamming distance* is one method which can be used to compute a matrix. It uses the sequence alignment to compute the different positions between the aligned sequences and generates a distance matrix. The distance matrix is then used as input for the distance-based methods. There are three main algorithms, namely *UPGMA*, *neighbor-joining*, and *FastME*.
The character-based methods are said to be the more reliable and informative methods, because they are able to "generate a full evolutionary history" (Jäger, 2013, p. 1). The raw data needs to be classified by character classes and this classification is not always available (Jäger, 2013). In contrast, distance-based methods are popular for handling many different types of data. Therefore, they can also be used to represent data which can not be classified by character classes. Another advantage of the distance-based methods is their efficiency. The methods can handle large data sets in a short time. Distance matrices can represent thousands of taxa and the distance-based methods can reconstruct trees using this large distance matrix in a small amount of time (Huson, Rupp, & Scornavacca, 2010; Jäger, 2013). Within linguistics, both kinds of methods are used. Nevertheless, for this project one of the distance-based methods is chosen because of the large data set and the efficiency that comes along with it.
Before we come to the advantages and disadvantages of the chosen method over the others, I want to introduce the three main algorithms.
UPGMA is the abbreviation for *unweighted pair group method using arithmetic averages* and it is the oldest method of the three described here. It is a distance-based method, therefore the input data is a distance matrix. The output of the method is a rooted tree and the edges of the tree have a specific length. The algorithm of the method clusters the given data and merges two clusters at each stage while creating a new node in the tree at the same time (Huson, Rupp, & Scornavacca, 2010). The tree is build bottom-up, which means the method first clusters pairs of leaves and then pairs of clustered leaves and so on. Each node receives a height and the difference between the heights of two nodes is used to

compute the length of the edge (Huson, Rupp, & Scornavacca, 2010). The result is a rooted, *ultrametric* tree. An ultrametric tree is a tree where every node has the same distance to the root.

Neighbor-joining can be seen as a modification of UPGMA and it is the most popular method of the three methods introduced here. This method computes an unrooted tree with edge lengths out of a distance matrix. The algorithm clusters the given data and computes the average distance between two clusters to all other clusters (Huson, Rupp, & Scornavacca, 2010). This is done to avoid the effect of large distances between two clusters which are actually neighbors in the tree. Therefore, the correct neighbors can be computed. This is a debility in the UPGMA algorithm where the nodes with the smallest distance are always neighbors. Additionally, this method avoids also the effect of an ultrametric tree. Within the algorithm, a *neighbor-joining matrix* is created. The clusters with the minimum entry in the matrix are paired and a new pair of neighbors arise (Huson, Rupp, & Scornavacca, 2010). One cluster represents one node in the resulting tree.

FastMe is the newest distance-based method of the three and it is developed within the *balanced minimum evolution (BME)* framework. The method computes a binary branching tree with the help of a distance matrix. The *balanced average distance* is the distance between two taxa which are represented by two nodes. The distance is called *balanced* because the two taxa have an equal weight at the beginning of the calculation. This average distance is used to compute a *balanced edge length* which is assigned to every edge in the tree (Huson, Rupp, & Scornavacca, 2010). Finding the optimal tree with this method is an NP-hard task. Therefore, heuristics to compute an BME tree need to be taken into account. The FastME heuristics is based on two phases. Within the first phase, an initial tree is created and in the second phase, the tree is improved using *nearest neighbor interchange (NNI)* operations until no more improvements can be made. An NNI operation swaps subtrees which are attached to the same edge in all possible ways. Additionally, the NNI operation finds iteratively the minimum entry within a neighbor-joining matrix (Huson, Rupp, & Scornavacca, 2010). Desper and Gascuel (2002) provide the software package *FastME* to reconstruct trees. It contains different algorithms to compute the initial tree and different algorithms to improve the tree via tree swapping. The software package can be found under `http://www.ncbi.nlm.nih.gov/CBBresearch/Desper/FastME.html`.

Jäger (2013) uses the FastME software package to find the optimal combination of the implemented algorithms. There are five algorithms to reconstruct the initial tree and three methods for post processing (Jäger, 2013). Jäger (2013) uses linguistic data from the ASJP database and tests each combination of the algo-

rithms. Two variants of the neighbor-joining algorithm, *BIONJ* (an improved version of the neighbor-joining algorithm) (see Gascuel (1997a)) and *unweighted neighbor joining (UNJ)* (see Gascuel (1997b)), produce the best results for reconstructing the initial tree. The *ordernary least square NNI* improves the results of BIONJ and UNJ during post processing. Jäger (2013) shows a comparison of the implemented algorithms within the FastME package and gives an evaluation of the results. This paper is helpful to decide which implemented algorithm to use to reconstruct trees with FastME.

3.3 Reconstructing Linguistic Trees

Phylogenetic trees in linguistics are trees computed my means of linguistic data and their labels correspond to languages. To create such linguistic trees, a linguistic database containing information about different languages and a phylogenetic reconstruction algorithm are needed. The linguistic database is the ASJP database and the reconstruction algorithm used is BIONJ from the FastME software package.[1] A distance matrix is computed for either all languages or a subset of languages included in the ASJP database.[2] The alignment for the calculation of the distances is computed by the*Needleman-Wunsch algorithm* (Huson, Rupp, & Scornavacca, 2010). This algorithm estimates a global alignment of the phonetic representation of two words. These alignment scores are aggregated in a 40×40 matrix where at position (i, j) the scores for the i-th word of language one and the j-th word of language two is stored. Using this matrix the distance between the respective languages can be calculated. This distance is estimated by the *Distance based on Corrected Evidence of Relatedness (dERC)* introduced by Jäger (2013). The dERC algorithm exploits the idea that words from related languages describing the same concept are more similar than from unrelated languages. In the 40×40 matrix, words describing the same concept are stored in the diagonal. In a distance-based approach, high similarity means low distance. Therefore, for related languages the values along the diagonal are assumed to be smaller then the off-diagonal values whereas for unrelated languages this should not be the case. This idea can be formalized by ranking the different normalized scores for two languages and calculating a maximum likelihood estimator from this ranking. This ranking is of course influenced by the number of missing entries, since they affect the ranking of the scores. To minimize the influence of the number of missing entries, the maximum likelihood estimator gets abstracted

[1] In the following, FastME (algorithm) is short for combination BIONJ ordernary least square NNI.

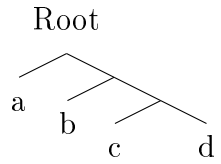
[2] Thanks to Gerhard Jäger, for providing the python code for the computation.

from the number of concepts.[3] These values can be easily converted into distances. These distances are then stored in a |language| × |language| matrix. The distance matrix can be used to create a phylogenetic tree with the FastME algorithm.[4]

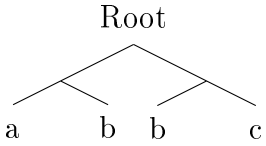
There are two types of linguistic trees which are created for our purpose. First I will explain their counterpart in phylogenetics and will then describe the linguistic trees.

In phylogenetics, rooted trees can be used for representing two main concepts, namely *species trees* and *gene trees*. The evolutionary history of an organism is represented in a species tree, whereas the evolutionary history of its genes is represented in a gene tree.

(14) a. species tree:



b. gene tree:



Organisms can have a different evolutionary history than their corresponding genes. Therefore, the trees which represent the histories can also differ, as can be seen in (14). During the evolution, evolutionary events take place. These evolutionary events can be a duplication, loss or transfer of genes. Caused by these evolutionary events, the genes can have a different history than their organism and therefore the gene trees may differ from the species tree.

Gene and species trees can be used to represent the relation of two or more organisms to their ancestor. Within the species tree, the ancestor and its evolutionary history will be illustrated. Inner nodes of the species tree sketch the speciation of the descendant organism(s). The evolutionary history of the genes of the descendant is depicted in a gene tree. For each descendant a gene tree can be reconstructed. For a comparison of their history the gene tree can be mapped to the species tree or vice versa (Huson, Rupp, & Scornavacca, 2010). Additionally, evolutionary events of the history of genes of one descendant can not be detected within a species tree, but only by mapping the gene tree to the corresponding species tree. If we have more than one gene tree, we can use two different methods for comparison. The first is mapping each gene tree to the species tree. The other method is to map two or more gene trees to each other and to merge into a

[3] For the mathematical details see Jäger (2013, p. 248 ff.)

[4] Thanks again to Gerhard Jäger for providing the python code.

network to represent their common history. This network can then be mapped to a corresponding species tree to compare their history. This method is used for the comparison of speciation events and period of the speciation of their descendant organisms (Wiley & Lieberman, 2011).

Languages can have a different evolutionary history than their words. Therefore, the history of languages can be represented in a tree and the history of words, according to their language, can be represented in another tree. This is similar to the distinction of gene and species trees of phylogeny. The species tree would represent the language history and the gene tree the history of words. It can be said that a language is a linguistic species and a word is a linguistic gene. To avoid confusion, I will further refer to linguistic species trees as *language trees* and to linguistic gene trees as *concept trees*.
The computation explained above is the same for the reconstruction of language trees, as well as for the reconstruction of concept trees. The output trees of FastME are all binary branched and unrooted. The trees are then rooted with respect to an outgroup. This outgroup is a language or phonological representation which is located outside of a main group to which it is closely related.
Jäger (2013) shows, that the automatically created trees by FastME are almost as good as the manually created trees by experts. The automatically created trees show a great fit to the manually created trees which proves that FastME produces similar trees than Experts. Therefore, the linguistic trees can be used for further studies and as input to other phylogenetic methods.

3.3.1 Language Trees

A language tree is a tree which represents the history of different languages. The set of all languages can either be all languages contained in the ASJP database, or all languages contained in one language family or a set containing selected languages. A tree which represents all languages contained in the ASJP database is very large, because the database includes many languages from different language families. Therefore, the whole language tree cannot be displayed in the appendix. Nevertheless, the language tree for the Indo-European languages is represented in appendix D. Please note, that the tree is too large to fit on one page. Therefore, it was split in the middle for reasons of readability.
A language tree represents a set of languages. All languages are represented by the same set of concepts. A set includes the 40 concepts of the ASJP word list. The list can be found in appendix C. It might be the case that one or more concept representation is missing for a specific language. This might be due to

the fact that either the language has no representation for this meaning or that it is just not translated. The missing entries are marked, therefore the set of words stays the same and does not have a great impact on the computation. However, a solution to the problem of missing entries is stated below. The language set is used to compute a distance matrix with the dERC algorithm and the distance matrix is used for the reconstruction of the tree with the FastME algorithm.
The tree in figure 3.1 represents a set containing only Germanic and Romance languages.

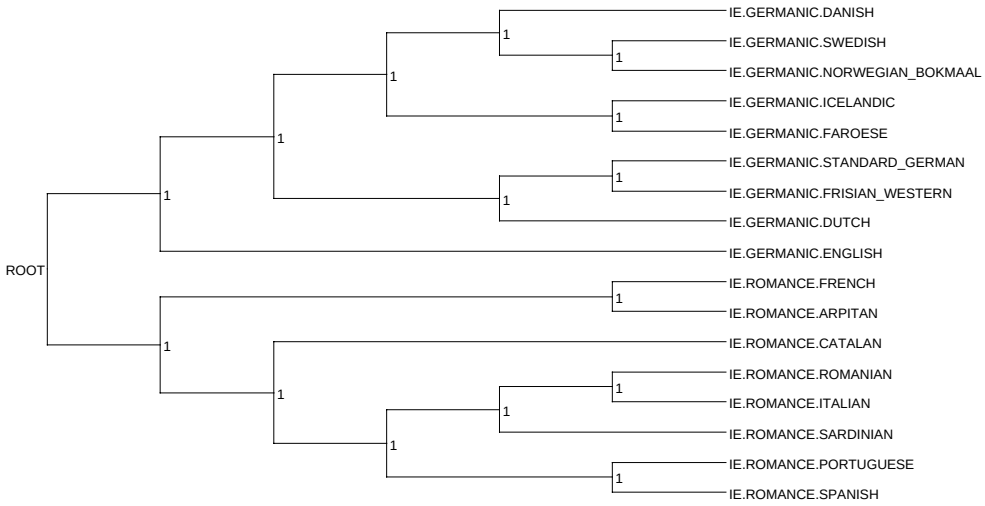


Figure 3.1: The language tree of Germanic and Romance Languages

The two language families are grouped as expected. The Germanic languages form one group and the Romance languages the other. Additionally, the grouping within the families is also as expected. Within the Germanic family, the Scandinavian languages, Danish, Swedish, and Norwegian form one group. Icelandic and Faroese form another which is clustered with the Scandinavian languages. The next group is German and Frisian, then Dutch and then English. Within the Romance family, French and its dialect Arpitan form one group. Spanish and Portuguese form another, while Italian and Romanian are sisters and a sister to Sardinian. This whole group is a sister to Catalan. The tree is binary branched and rooted. The tree is not rooted on an outgroup, because there are only two language families present. They split as expected at the root.
An example of a tree which is rooted on an outgroup is shown in figure 3.2.
The tree is reconstructed on a language set containing different Germanic and Romance languages. It is the same set of languages used above including two additional Austronesian languages. The grouping of the different languages is as expected. The two Austronesian languages can not be included in neither of the groups and serve therefore as the so called outgroup. The tree is rooted on this

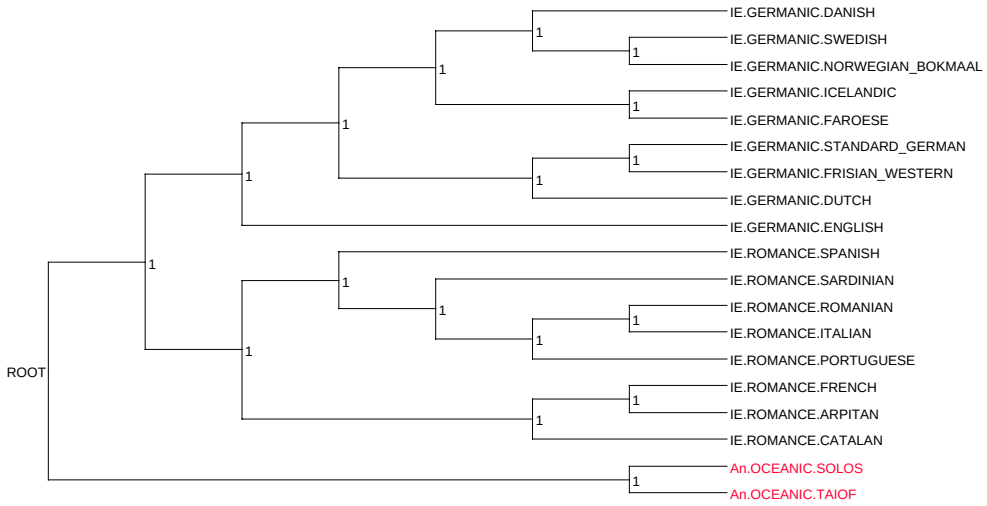


Figure 3.2: A language tree rooted on an outgroup

outgroup. In other words, the outgroup is one child of the root and the group of all other languages is the other.
Mostly, there is only one language tree which represents a specific set of languages with their corresponding words. This is not the case for concept trees, as can be seen in the following section.

3.3.2 Concept Trees

A concept tree represents the history of a single concept within different languages. The above claimed that a word can be seen as a linguistic gene, as stated, the term *word* is replaced by the term *concept*. A concept is a representation of a specific meaning within a language and it can be the case that a language needs more than one word to express this meaning. Nevertheless, concept trees can be seen as linguistic gene trees.
A concept tree illustrates the relation of a concept within a set of languages. In other words, a set of languages is given, one concept of the set is selected and a tree for this concept is constructed. The tree shows the history of this concept within the given set of languages. The concept is one of the 40 concepts represented in the 40 words list of the ASJP database. If a concept is missing within a specific language, the missing entry will be included in the reconstruction process. This can cause problems in the reconstruction. If there are one or more languages with such a missing entry, the expectation would be a group containing all languages with missing entries which serves as an outgroup. However this is not the case, because the dERC algorithm is already aware of missing entries. The missing entries do not have a great impact on the result of the distance matrix, they actually have “a positive effect”. The algorithm already corrects the

impact of missing entries and this is exactly what the tree in figure 3.3 shows. The tree is reconstructed using a set of Germanic languages and the concept "you". The red labels mark the languages which do not have an entry for this concept. Additionally, the concept representations for each languages are given in brackets behind the language name.

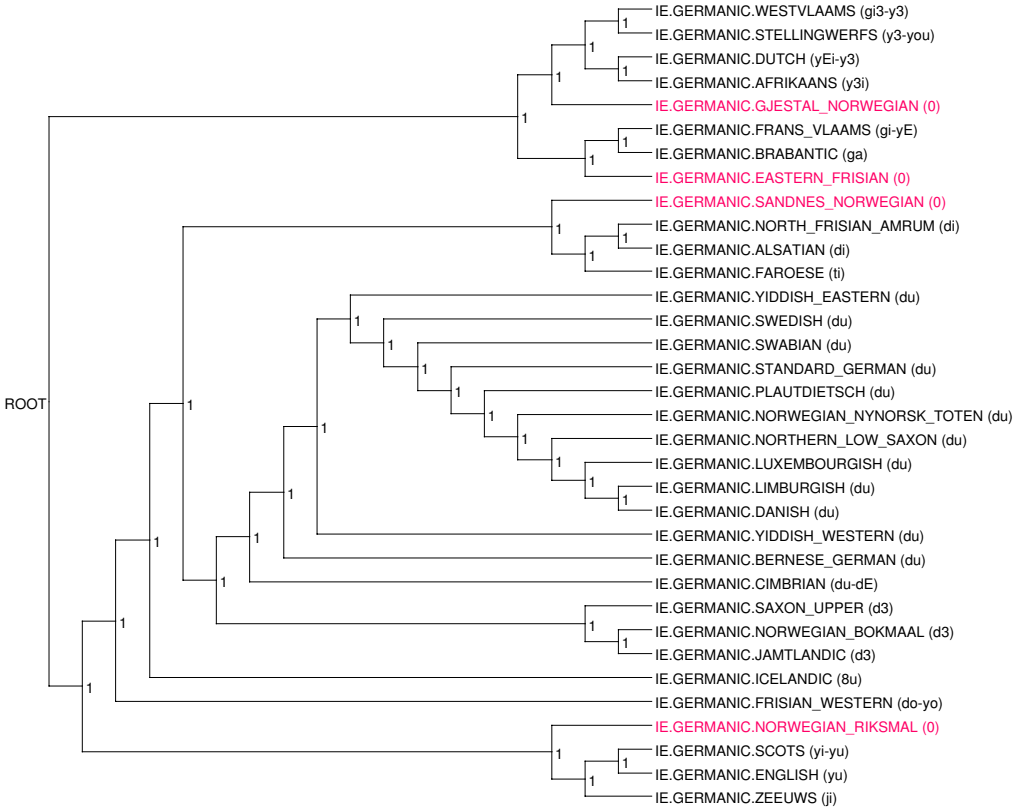


Figure 3.3: A concept tree with the missing entries

The missing entries are more significant for concept trees than for language trees. Within language trees, missing entries do not have a great impact on the results, because all other concepts can be used for the reconstruction. For concept trees, missing entries have a greater impact on the results, because one cannot be sure if they are grouped correctly within the tree. This might cause problems while mapping concept trees to language trees. A solution to this problem is stated in the next section.

There can be as many trees as concepts. In our case, there can be 40 concept trees because the ASJP list contains 40 different concepts. Given a set of languages, there can be one language tree representing the history of the languages and 40 different concept trees representing the history of each single concept. Since each concept can have a different history, the grouping of the languages within the tree differs. Therefore, each concept needs is own computation of a distance matrix and concept tree. The concept representations within the language sam-

ple are used to compute a distance matrix with the dERC algorithm stated in (Jäger, 2013). Every different distance matrix illustrates the distances between the languages for the corresponding concept. The distance matrix is used for the reconstruction of the corresponding tree with the FastME algorithm. The result is a concept tree representing the relation between languages for a single concept. The tree in figure 3.4 represents the grouping of Germanic and Romance languages only using the concept "mountain". The languages remain the same as in figure 3.1, additionally the phonetic representation of ASJP is given in brackets behind each language name. This tree represents the history of the concept "mountain" for a specific language sample. The grouping differs from the one in the language tree in 3.1. This is due to the fact, that concepts can have different histories than languages. These differences can only be revealed by a concept tree. According to the phonetic representations, the grouping is as expected. All similar representations are grouped together.

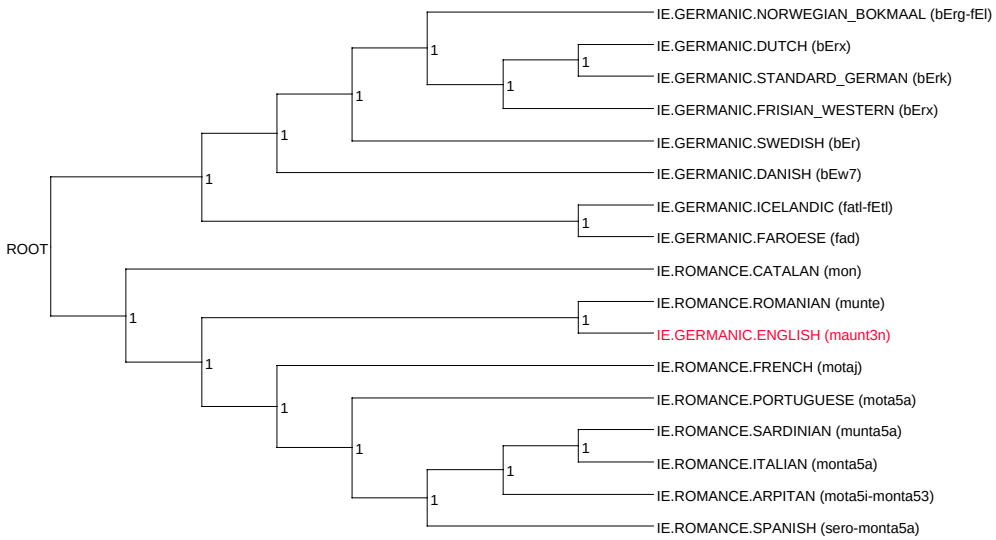


Figure 3.4: The concept tree for mountain of Germanic and Romance languages

The interesting language here is English. The English word *mountain* is a loanword from old French *montaigne* (Joseph & Janda, 2003). All other Germanic languages use another stem for the word, like *berg* or *bjerg*. Therefore, the concept tree of "mountain" in figure 3.4 has a different grouping than the language tree in figure 3.1. The history of the word comprises an evolutionary event, namely borrowing, which cannot be detected in the language tree. Clearly, such evolutionary events within the history of words can be detected by a comparison.
The comparison of language and concept trees should give a better insight into the different evolutionary histories. Therefore, different algorithms and methods need to be tested to see what it is that can be detected while mapping a concept tree to a language tree.

3.3.3 Improving linguistic trees

The ASJP database contains 40 concepts for over 200 languages, therefore it is not surprising that some languages lack one or more concepts. The missing concepts are marked to avoid different lengths of word lists for different languages. With word lists of different lengths, the distance matrices and the linguistic trees could not be computed with complete accuracy. Therefore, marking these missing concepts was the best solution provided by the ASJP database.
Nevertheless, for the reconstruction of linguistic trees, the missing concepts can cause problems. This can also affect the phylogenetic analysis of mapping the gene tree to its corresponding language tree. Therefore, the linguistic trees need to be improved by replacing the missing concepts.
For improving the linguistic trees, two new programs are implemented, one for language trees and one for concept trees.[5] The programs are compact and create a tree for a user specific language sample, by replacing the missing concepts. Both programs are almost identical, the difference consisting of the computation of the distance matrices and the linguistic trees. Both programs consist of four main parts, the following should provide you with a short overview:

(15) Language Trees:
   a. Replaces the missing concepts for the language sample
   b. Creates new PMI scores for the specific language sample
   c. Creates a distance matrix using the dERC algorithm
   d. Creates a language tree with FastME

(16) Concept Trees:
   a. Replaces the missing concepts for the language sample
   b. Creates new PMI scores for the language sample
   c. Creates 40 distance matrices using the dERC algorithm, one for each concept
   d. Creates 40 concept trees using FastME, one for each concept

The main focus of the program was to implement a way for replacing the missing concepts. The *Partial Mutual Information* (PMI) score $PMI(a, b)$ is a measure for the probability that the two segments a and b evolved along different phylogenetic branches from the same ancestor (Jäger, 2013). This gives us a "graded notion of similarity between sounds" (Jäger, 2013, p. 263) which is necessary for weighted string alignment.[6] The distance matrices are computed in the same way

[5] The program is not available online. If you are interessted in it or would like to use it for your research, please feel free to contact me.
[6] Thanks to Gerhard Jäger, who provided the python code.

as was explained in the previous sections. The same holds for the computation of language and concept trees. The core was not only the implementation of an algorithm to replace the missing concepts, but also to create an overall program which should facilitate the creation of linguistic trees.
The only input, which needs to be provided by the user, is a file including the longnames of its language sample. The longnames are the language classifications used by Dryer and Haspelmath (2013). Each language has its own line and the order needs to be the same as within the ASJP Matrix or the file containing every language within the ASJP database. Of course, these files are contained within the program so the user can use them for creating its own language sample. Such a file could look like this:

(17)
IE.GERMANIC.DANISH
IE.GERMANIC.DUTCH
IE.GERMANIC.ENGLISH
IE.GERMANIC.FAROESE
IE.GERMANIC.FRISIAN_WESTERN
IE.GERMANIC.ICELANDIC
IE.GERMANIC.NORWEGIAN_BOKMAAL
IE.GERMANIC.STANDARD_GERMAN
IE.GERMANIC.SWEDISH
IE.ROMANCE.ARPITAN
IE.ROMANCE.CATALAN
IE.ROMANCE.FRENCH
IE.ROMANCE.ITALIAN
IE.ROMANCE.PORTUGUESE
IE.ROMANCE.ROMANIAN
IE.ROMANCE.SARDINIAN
IE.ROMANCE.SPANISH

This is the Germanic and Romance language sample used for computing the trees in figure 3.1 and 3.4. Afterwards, the program computes everything necessary to construct a language or concept trees.
The crucial things needed for the comparison are all provided, for example the overall ASJP Matrix containing every language and its corresponding word list, the 41 sounds used within the database (they can also be found in appendix B) and the language tree of all languages contained in the database. Additionally, all files which are computed during the program are stored in an output folder. This is the ASJP matrix of the language sample, the newly computed PMI scores, the

ASJP matrix with the replaced concepts, the distance matrix (or the 40 distance matrices) and the language tree (or the 40 concept trees). The user can use every output for further analysis.
As stated above, the main part of the program was the implementation of an algorithm to replace the missing concepts present within the language sample. I want to explain this part in more detail.
The main idea of replacing the missing concepts is to search for the closest related language and if the concept is present there use it, otherwise continue to search for the closest language and its concept representation. The algorithm is successful if all missing concepts are replaced within the ASJP matrix of the language sample.

First of all, the missing concepts need to be found within the ASJP matrix of the language sample. All missing concepts are stored in a list. This list contains the index of the language within the ASJP matrix, the name of the language and the index of the missing concept. The language tree which contains all languages represented in the ASJP database is read. I choose the language tree which is computed in the same way as the language trees explained above. This is due the following three reasons. First, I need a so called expert tree to find the closest related language. Usually, an expert tree is a tree created manually by linguists. Nevertheless, (Jäger, 2013) stated that the tree computed for all languages within the ASJP database using the FastMe algorithm is similar to the expert trees. For example, (Lewis, 2009) provides such an expert tree within Ethnologue. The classification of the tree within the newest version (Seventeenth edition) and the classification of the language tree created by FastMe are highly similar, as stated in (Jäger, 2013). Therefore, the language tree computed by FastMe can serve as expert tree. Second, the languages represented in the language tree are the same as within the matrix. In other words, we need to make sure that all languages within the user specific language sample are present within the language tree. Third, the languages need have the same spelling to be able to find the right language and its closest related language.

In the next step, the closest related language(s) and their concept representation(s) need to be found and saved. We search for each language, which has a missing concept, the corresponding node in the expert tree.

(18) STANDARD_GERMAN IE GERMANIC 83812810 iX du vir ains cvai mEnS fiS hunt laus baum blat haut blut knoX3n horn 0 aug3 naz3 can cuN3 kni hant brust leb3r triNk3n ze3n her3n Sterb3n kom3n zon3 StErn vas3r Stain foia pat-vek bErk nat fol noi nam3

(19) ARPITAN IE ROMANCE 137000 Z3-c3 t3 no-nu yo dow omu p3so Si-ci 0 abro foL3-foLi 0 so-sok 0 0 or3L3-or3Li yi na-no dE lEwa-lEga Z3nu-c3nu ma 0 0 bEa vEa ekota-exota mori-mo8i v35i-v3ni solEL-sElol etela-e8ela Ewa-Ega pEri-pE8i 0 Sami-cami mota5i-monta53 nE-nE plE-pE novo-nu no

Example (18) illustrates the entry in the ASJP matrix for German. German is missing one concept, which is colored red. The missing representation is the concept for "ear". In Example (19), the entry for the French dialect Arpitan is given. This language misses seven concept representations. To replace all missing concepts, the concepts of the most closely related language need to be found.
To get the most closely related language, we get the sister of this node. The sister can either be a leaf, which represents one language, or it can be an inner node, which has more than one descendant.
If the sister node is a leaf and therefore represents only one language, the word list is searched for the concept. In figure 3.5, the section of the Romance languages within the expert tree is illustrated. Arpitan (coloured in blue) is the language with the missing concepts. Its direct sister node is French (coloured in red).

Figure 3.5: The section of Arpitan and French in the expert tree

The language can only serve as a possible candidate if the concept has a representation in the word list. If this is the case, the closest related language and its concept representation are saved.

(20) FRENCH IE ROMANCE 68458600 j3 ti-vu nu oe de om paso Sia pu arbr3 f3y po sa os korn ore 3y ne da lag j3nu ma patrin fa ba va otadr

muri v3ni sole etol o per fe rut motaj nui pl3 nuvo no

The example in (20) shows the ASJP entry for French. French has a concept representation for every missing concept in Arpitan (coloured red). Therefore, the missing concepts in Arpitan can be replaced by the French concept representations.

If the sister language does not contain a concept representation, the tree is searched iteratively and bottom up from the node (the language with the missing entry) for the next related language. It only stops, if one or more related languages are found which provide a representation of the missing concept. If the next related language is a leaf node, the procedure is repeated. Otherwise, the following strategy is necessary to find the best concept representation.

The sister node of the language with the missing entry can also be an inner node. This means that the node can have one or more descendants. In figure 3.6, the section of German and its sister node with the corresponding descendants is illustrated. German (coloured in red) is the language with the missing concept. Its sister node is an inner node (coloured in green) and its corresponding descendants (coloured in blue) are possible candidates from which the missing entry can be taken.

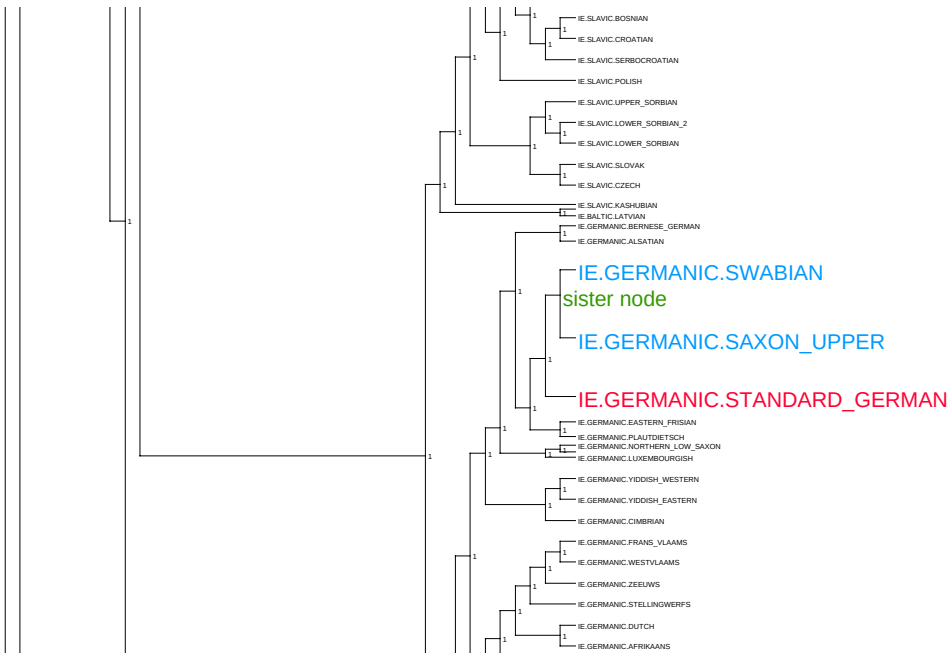


Figure 3.6: The section of German and its sister node in the expert tree

(21) SWABIAN IE GERMANIC 819000 i du mia ois cvoi mEnZE fiS hund laus bom blad haut blud knoXE hoan EalE augE-oiglE ciNgE-nas can cuN-cuNE knui hEnd buSt leba diN se hea StEab-Stuab kom sonE StEan vaza-voza Stoi fuia veg bEag nad fol-ful noi nom-nomE

(22) SAXON_UPPER IE GERMANIC 2000000 iS d3 mia ens cve mEnS fiS hunt lous bom plot hout pl3t knoN hoan 0 oX3-gukS3 noz3 con cuN3 kni hont-fod3 pXust lEba tXiNkN-kudln zen-kukN hean StEam kom zon3 StEan vosa Stoin foia fot-veS bEak not fol noi nom3

The algorithm looks for each descendant language that has a representation for the concept and saves the information. It could be the case that only one descendant has the corresponding concept representation, like in the examples (22) and (21). Only Swabian has a representation for "ear" and Saxon doesn't. Therefore, only Swabian is taken into account and the concept representation is saved and replaced later. If no descendant has a representation, the tree is again searched iteratively and bottom up from the node (the language with the missing entry) for the next related language(s) until one or more languages are found which have a representation for the concept. It could also be the case, that all descendants have a concept representation. Therefore, a method is needed to filter out the best candidate or the closest related language from all possible candidates.

(23) SARDINIAN IE ROMANCE 1045180 deo 0 nos unu duos pesone piSe kane priogu arvure foza pede sambene osu koru uriga ogu nasu dente limba enugu manu petus figadu biere biere intendere morere benere sole isteda aba pedra fogu istrada munta5a note prenu nou lumene

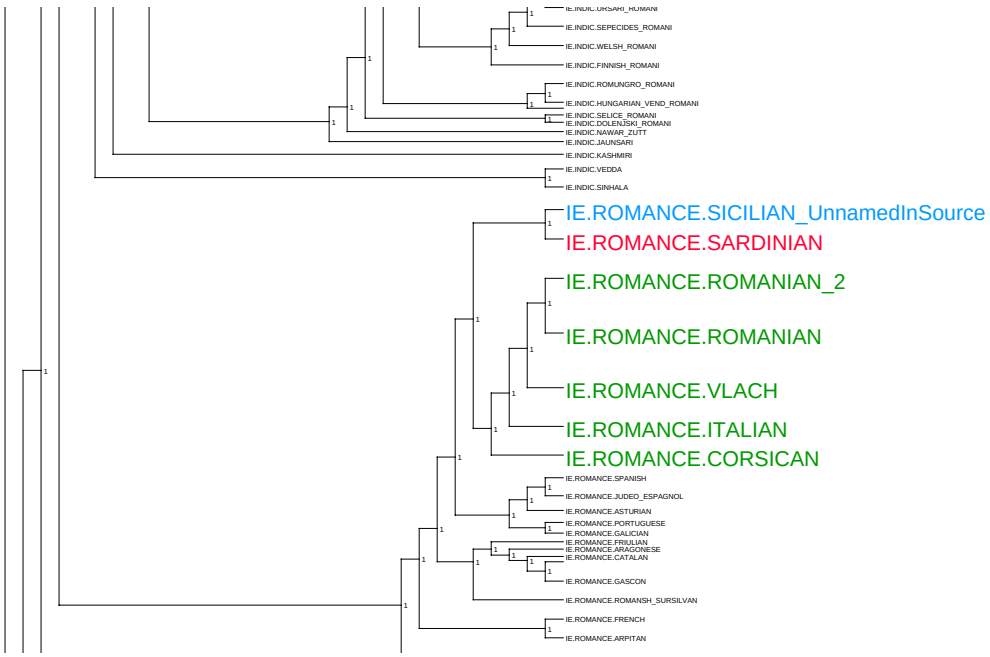


Figure 3.7: The section of Sardinian and its sister nodes in the expert tree

In example (23), the ASJP entry of Sardinian is listed. In figure 3.7, the direct sister to Sardinian is Sicilian. Therefore, the concept representation of "you" is

searched within Sicilian.

(24) SICILIAN_UnnamedInSource IE ROMANCE 4700000 iu-eu 0 nuautri unu du-dui pirsuna peSi-piSi kani 0 arbulu fogia pedi saNgu osu kornutu orikia okiu naska-nasu denti liNa dinokiu manu petu-mina fiCatu biviri-vippita vidiri sentiri-ascutari moriri-muriri veniri-viniri suli stida aka petra foku strata-via munta5a noti Cinu-saciu novu nomi-nomu

As presented in example (24), Sicilian doesn't have a concept representation for "you" either. Therefore, the algorithm goes one node up and searches for the sister node. The sister node is an inner node. All descendants which have a concept representation for "you" are saved in a list. A method is needed to filter out the best candidate which can serve as closest related language. All the information about the language with the missing entry, the index of the missing concept in the word list, the closest related languages and their concept representations, are stored in a list. This list can contain one closely related language, but is can also contain a list with all possible candidates.
To filter out the best candidate from a set of possible candidates, the languages which serve as possible candidates are aligned to the language with the missing entry to find the most similar. For the alignment, the whole concept list of the languages is used and each possible candidate is aligned to the language with the missing entry. The alignment returns a similarity score and the highest score indicates the best alignment and therefore the best candidate or the closest related language.
In the case of Sardinian, figure 3.7 shows that the sister to its parent node contains more than one languages. All of the five languages are aligned to Sardinian and a similarity score is calculated. Example (25) lists the languages, the corresponding concept of "you" and the similarity score. Italian has the highest similarity score and is therefore the nearest related language to Sardinian. The concept representation of Italian is used to replace the missing concept in Sardinian.

(25)

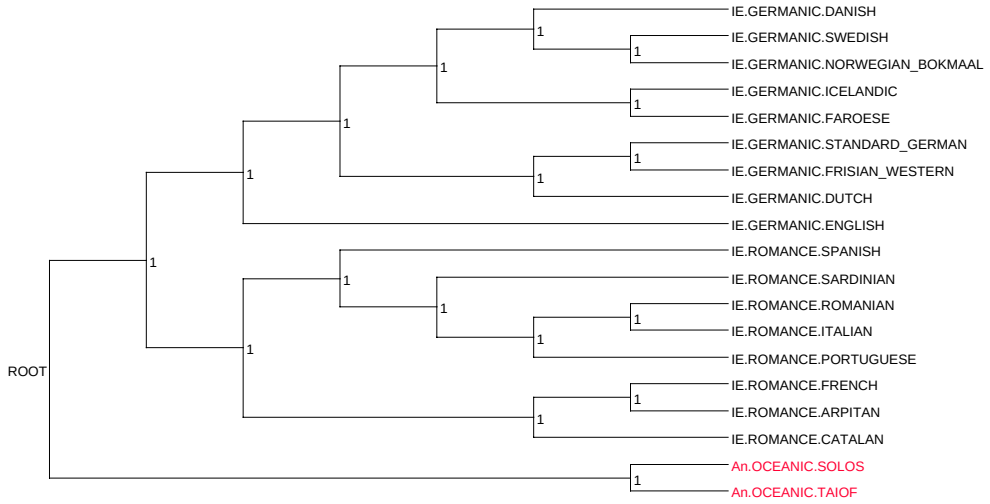
[['CORSICAN', 'tu', 24.741487675242027],
['ITALIAN', 'tu', 26.522553046371318],
['VLACH', 'tini', 16.622748237147697],
['ROMANIAN_2', 'tu', 21.828200439852459],
['ROMANIAN', 'tu', 26.20017747760927]]

After the alignment, we have a list containing the language with the missing entry, the index of the language, the index of the concept and the concept used for

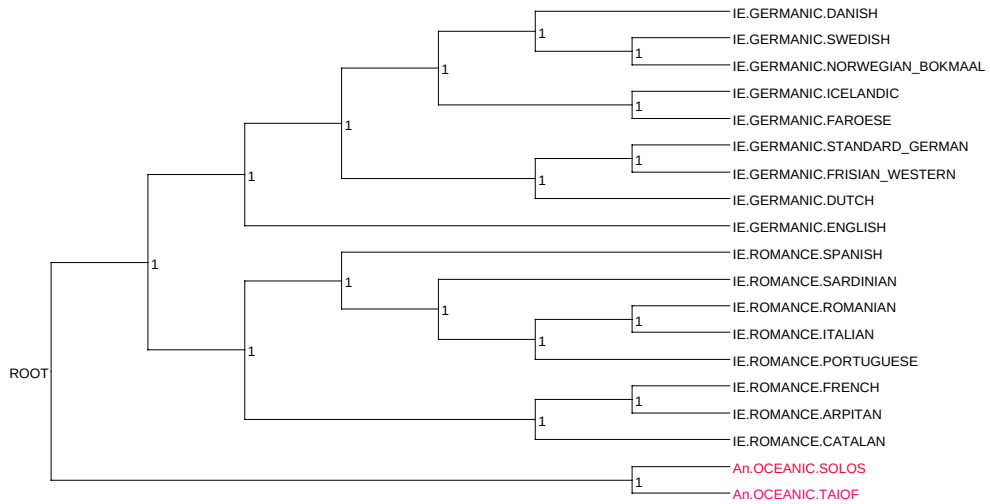
the replacement. Within the replacement function, all languages with a missing entry are checked in the ASJP matrix for the language sample and the missing concepts are replaced with the concepts from the closest related languages. The ASJP matrix for the language sample is updated and used for the next analysis. The pseudocode for the most important methods are stated in appendix F.
With this algorithm, all missing entries within a language sample can be found and replaced. The idea of replacing the missing concept with one concept of a closely related language is intuitive. The ASJP database contains more than 200 languages, so it can be supposed that the concept of a closely related language is similar to the actual concept representation. Additionally, the method replaces the missing entries automatically. This is a faster way of improving the data and also the linguistic trees than replacing the missing concepts manually.
After replacing the missing concepts and updating the sample matrix, the next step in the program is the computation of the distance matrix and the tree. For the language tree a single distance matrix is computed with the dERC algorithm. The newly computed PMI scores are used and a distance matrix is computed. This matrix is the input for computing the language tree. The script for the computation of the tree is also modified and adapted into the program. The FastME algorithm is used for the computation of the language tree. All files are saved for the user in an output folder. The only difference for the computation of concept trees is the calculation of the distance matrix and the tree. All 40 distance matrices are computed at once (one for each concept). Additionally, the tree is computed with the FastME algorithm, all 40 concept trees at once. Again, all files are saved for the user.

The question is, does the replacement of the missing entries improve the linguistic trees?
To answer this question and show the improvements of the trees, I will compare the trees displayed above with trees computed by the program.
The first comparison will be that of the language trees. I stated above that the missing entries do not have a great impact on the computation of the languages trees. This is due to the fact that the language tree is computed using all concepts provided by the database. In other words, the language tree is computed over all languages (in the sample) using all concepts. There is no language, which only consists of missing entries. The small amount of missing entries seen over all concepts from all languages does not carry much weight in the computation. The alignment of the other concepts compensate the missing entries. Therefore, the language tree with the missing concepts is the same as the language tree without the missing concepts. This can also be seen in the figures 3.8(a) and 3.8(b).

The comparison of the two trees shows the same clustering. The Austronesian languages are the outgroup in both trees, just as it was expected. The two other clusters are the Germanic languages and the Romance languages. The clusters are the same in both trees. Therefore, the replacement of the missing entries is no improvement to the language trees.



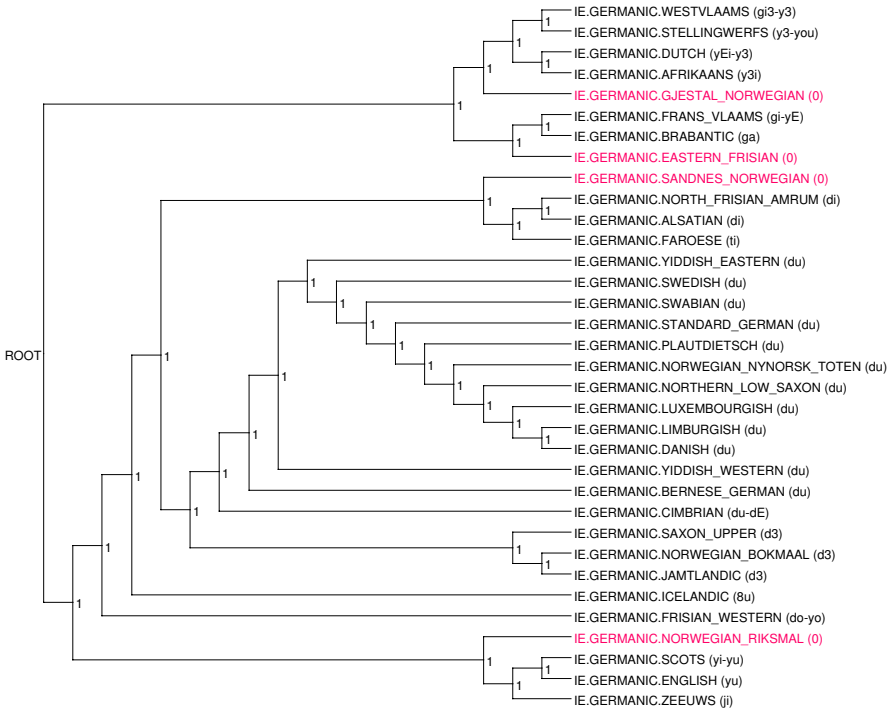
(a) With missing entries



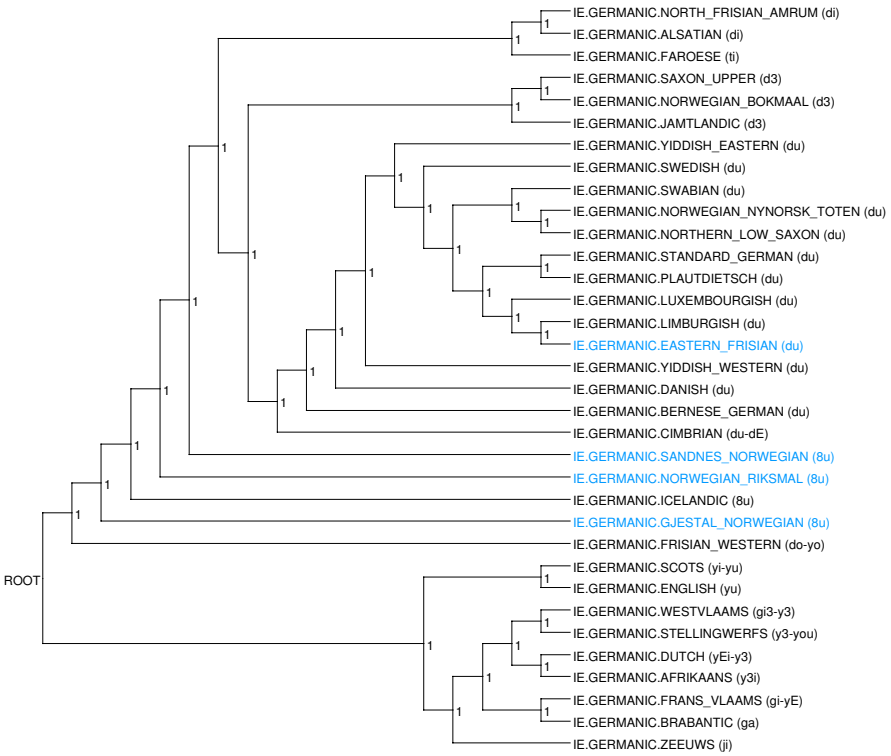
(b) Without missing entries

Figure 3.8: The language trees of the Germanic and Romance language sample with an outgroup

The comparison of the concept trees shows distinct results from the comparison of the language trees. I stated above that the missing entries do have an impact on the clustering of a concept tree. This can be seen in the comparison of the trees. In figure 3.9(a) the concept tree of the concept "you" shows a different grouping for all languages with a missing concept. The languages with a missing



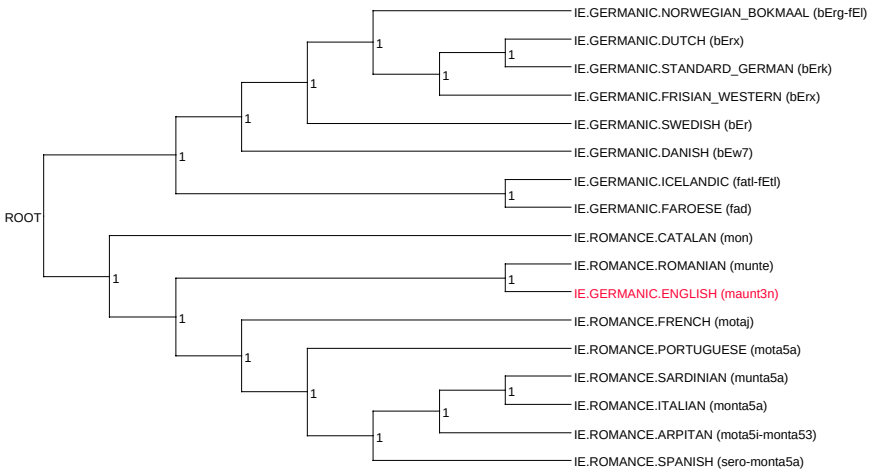
(a) With missing entries



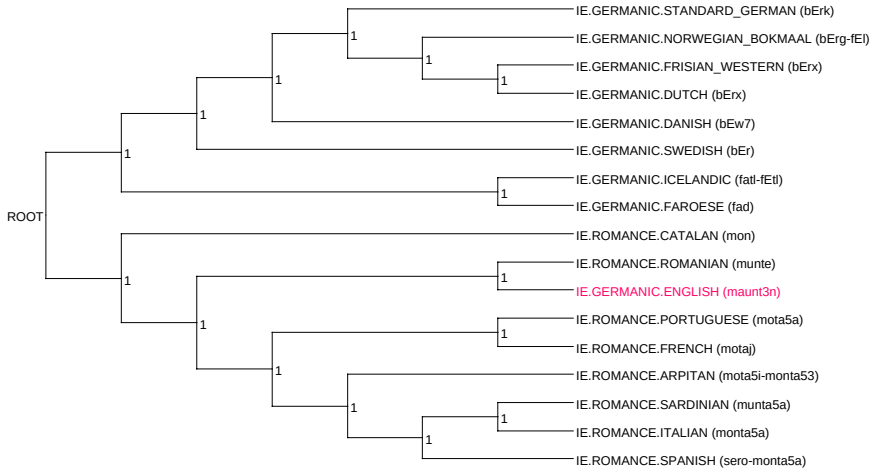
(b) Without missing entries

Figure 3.9: The concept trees "you" of the Germanic languages

entry don't constitute an outgroup. This is due to the algorithm. Although the dERC algorithm is aware of missing entries, it does not guarantee the correct



(a) With missing entries



(b) Without missing entries

Figure 3.10: The concept trees "mountain" of the Germanic and Romance language sample

grouping of the languages. However in the further analysis the different grouping affect the results. Therefore, we need to get rid of the missing entries. This is done by replacing the missing entries with the concept of its closest related language. Figure 3.9(b) shows the concept tree of the concept "you" for all Germanic languages with the replacement of the missing entries. All languages displayed in blue are the languages with a missing entry before the replacement and therefore had no representation for the concept "you". With the replacement, the languages acquired an entry for the concept and the tree is computed in a more intuitive way. Apparently the languages are inside their corresponding cluster and next to their closest related languages. The algorithm chooses the concepts of their closest related language within the expert tree. This concept is therefore present in the updated ASJP matrix and can be used to compute the concept tree. The improvement is observable within this tree. Within the

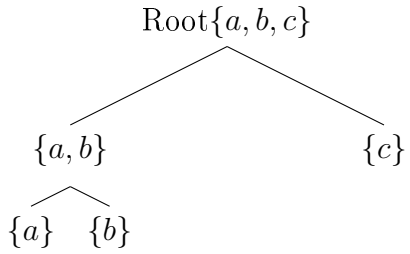
Germanic languages and the concept "you" there was more than one language with a missing entry. This is not the case for all language samples.
The comparison of the concept trees "mountain" in figure 3.10(a) and 3.10(b) shows that all languages have a representation of the concept "mountain". Therefore, there is no different grouping due to missing entries. However, there is a different grouping as a result of the computation. The English language can be found within the Romance languages in both trees, because English borrowed the word *mountain* from Old French. The different clustering within the Germanic languages is caused by the new computation of the concept tree. The program, which replaces the missing entries and computes linguistic tree, enhances these linguistic trees. The comparison between the concept trees in figure 3.9(a) and 3.9(b) shows this improvement needed for further analysis. To detect mismatches between trees, the concept tree can be mapped to its corresponding language tree. For this analysis, the concept trees should have concept entries for each language. Therefore, the newly computed concept trees without missing entries are better for the usage in such an analysis.

4 Comparing pyhlogenetic trees with Dendroscope

Dendroscope is a software which was implemented by Huson et al. (2007). The first version of the program was developed as "an interactive tool for drawing phylogenetic trees" (Huson, Rupp, & Scornavacca, 2010, p. 334). The idea was to implement a program which can display large phylogenetic trees. These large trees can contain up to 1 million nodes. In the second version of the program, Huson et al. (2007) extended the software to represent also phylogenetic networks. The software can also compute the networks from trees and display the results. The reason to implement a completely new program instead of using already existing ones was to create a better program which includes all necessary methods to display trees and networks. Huson et al. (2007) stated that the other programs always miss an important part, like the representation of large trees or more than one particular tree view. A comparison between Dendroscope and other programs shows that Dendroscope contains more possibilities to represent trees, as well as more possibilities to save the edited data. Additionally, the program provides a greater functionality than other programs.
In the third version of Dendroscope, Huson and Scornavacca (2012) provide methods to compute and visualize rooted phylogenetic networks. There are a number of already existing tools which can be used for computing unrooted networks, but there are only a few tools to compute rooted networks. According to Huson and Scornavacca (2012), these tools provide only limited use for biologist. Therefore, they implemented their own methods into Dendroscope (Huson & Scornavacca, 2012). This new version of Dendroscope includes algorithms to compute rooted phylogenetic networks out of rooted phylogenetic trees.[7]
As we saw in section 2, cladistics plays a central role in phylogenetics. A set of taxa is grouped together according to their common ancestor (Lecointre, 2006). This group is also called a *cluster* and clusters should "reflect the evolutionary history of a set of taxa" (Huson, Rupp, & Scornavacca, 2010, p. 127). Phylogenetic trees, especially rooted trees, display a set of clusters which is compatible. For example, we have a set of clusters C = $\{\{a\}, \{b\}, \{c\}, \{a, b\}, \{a, b, c\}\}$. These clusters are compatible and can be represented in one rooted tree.

[7] The newest version of Dendroscope can be downloaded from `http://ab.inf.uni-tuebingen.de/software/dendroscope/`

(26)



There can also be incompatible sets of clusters, which cannot be represented within a rooted tree. For example, the set of clusters C = $\{\{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}\}$ is incompatible, because their is no rooted phylogenetic tree which can represent C. This is the case, because b is connected with a and with c. To represent this set of clusters, a phylogenetic network is needed. These incompatible sets "may occur due to reticulate evolutionary events, such as hybridization or horizontal gene transfer, or they might reflect uncertainties due to insufficient data or inadequate analysis methods" (Huson, Rupp, & Scornavacca, 2010, p. 127).
There are two ways of computing a rooted network, either having a set of clusters or having a set of trees out of which a set of clusters is created. In our case, the rooted network is created out of rooted trees. The clusters represented within the trees form the set of clusters which is need to compute a network. By matching the set of clusters from trees, incompatible clusters may emerge. Those are represented within the networks.
A rooted phylogenetic network belongs to a group of networks which need to fulfill special criteria (Huson, Rupp, & Scornavacca, 2010):

(i) The network needs to be a directed acyclic graph (DAG)

(ii) The network needs one root

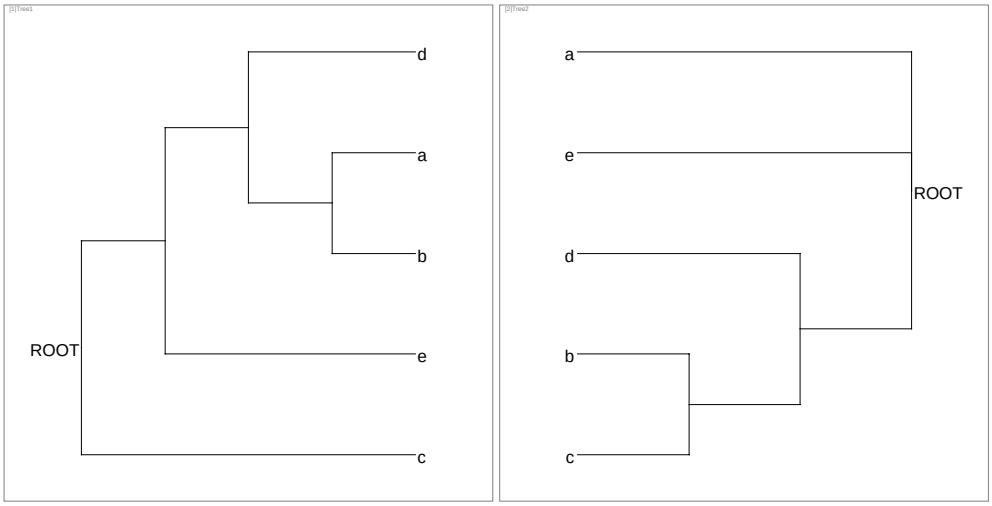
(iii) Each taxon is represented by a node

There are different kinds of rooted phylogenetic networks and for each network there is a method to compute it. The main approaches which are implemented in Dendroscope and explained in Huson, Rupp, and Scornavacca (2010) and Huson and Scornavacca (2012) are stated below. The input of all algorithms is one language tree and one concept tree. The trees are computed with a language sample of Germanic and Romance languages, which is also used above. The concept tree is of the concept "mountain" of the same language sample. The examples and the results are given in the following sections.

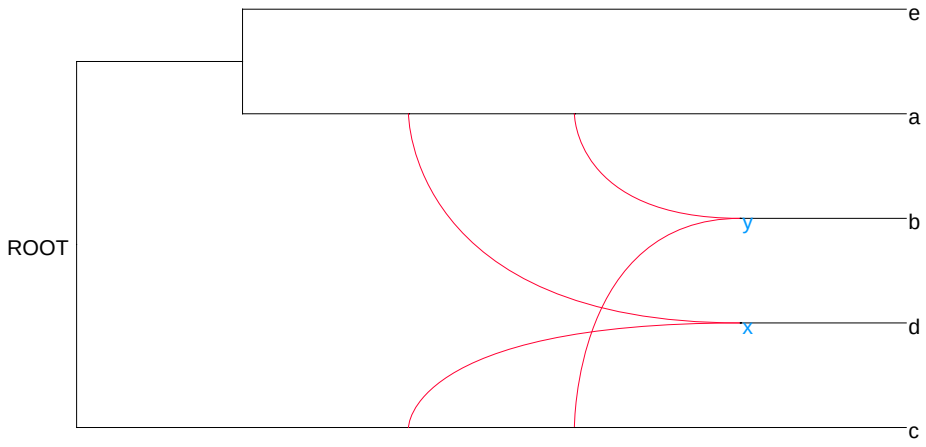
4.1 Cluster Networks

The computational method to compute a cluster network is an algorithm which compares trees with respect to their corresponding clusters. The trees can be

combined and the cluster network will represent the combination of the clusters within a network. The mismatches between the trees are represented by *reticulation nodes*. In figure 4.1(a) two trees are represented with 5 labels and different clusters. The result of a cluster network is represented in figure 4.1(b) and shows the matching of the two trees.



(a) Two trees



(b) A cluster network

Figure 4.1: A representation from two trees to a cluster network

In figure 4.1(b) the network is a directed acyclic graph (DAG). A DAG always has a root, which has an indegree $= 0$, nodes which have an indegree ≤ 1 and reticular nodes with an indegree ≥ 2. The indegree indicates the number of incoming edges. The root has no incoming edge, therefore the indegree $= 0$. A tree node has an indegree ≤ 1 which is mostly one incoming edge ending up in the tree node. A reticular node has an indegree ≥ 2 because it can have more

than one incoming edge. This can be seen in figure 4.1(b) where the reticular nodes are marked blue and the reticular edges are marked red. Huson, Rupp, and Scornavacca (2010, p. 134) differentiate between edge and reticular edge in the following way: "An edge $e = (v,w)$ is called a *tree edge*, if its target node w is a tree node and, otherwise, it is called a *reticular edge*". Additionally, he gives the following definition of a cluster network:

Definition 4.1 *Let χ be a set of taxa. A cluster network $N = (G, \lambda)$ on χ consists of a rooted DAG $G = (V, E)$, together with a bijective leaf-labeling λ: $\chi \to L(G)$.*

The pseudo code of the algorithm, implemented in Dendroscope, is stated in Huson, Rupp, and Scornavacca (2010) on page 135-136 with an additional illustration on page 137.
In phylogenetics, the typical application of cluster networks is the comparison of two or more gene trees for a set of species. The cluster network is the result which should illustrate the agreements and disagreements of the gene trees (Morrison, 2011). Nevertheless, a cluster network can also be used for comparing a gene tree and a species tree or in our case, a language tree and a concept tree.

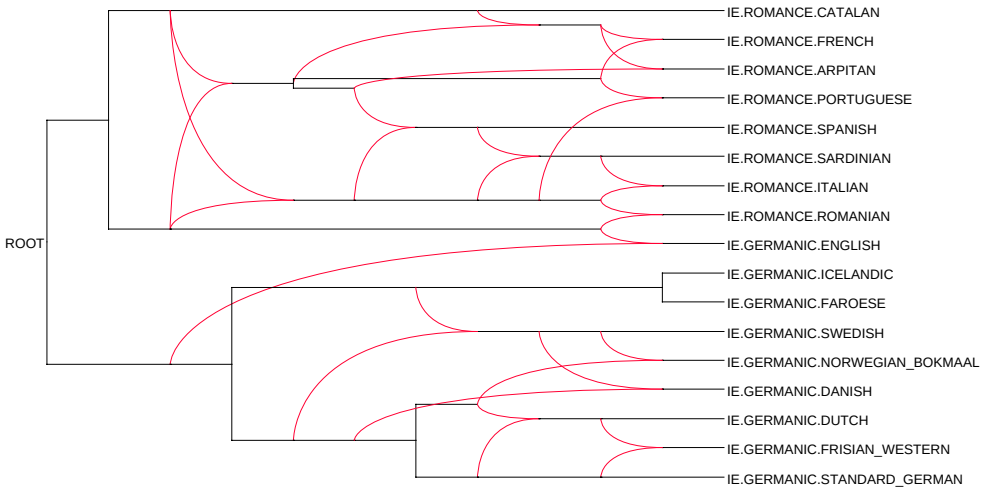


Figure 4.2: A cluster network of a language tree and a concept tree

There are a lot of disagreements between the language tree and the concept tree. The clustering corresponds most closely to the one of the language tree. In other words, the Germanic languages still form one group and the Romance language the other. English is in the middle, connected with the Germanic language family and additionally connected with Romanian, the language with which English is clustered in the concept tree. This is only one example. The reticular edges in the cluster network displayed in figure 4.2 are marked red. They indicate the

mismatches between the language tree and the concept tree. Morrison (2011) states that, the cluster network is used to represent the set of clusters and the mismatches between the clusters. The question arises, if all reticulation nodes can represent an evolutionary event like speciation, duplication-loss, or horizontal transfer. Morrison (2011) differentiate between cluster networks and the other networks which are described later on. He states that cluster networks are only used to display the mismatches and that the other networks are mostly used for an evolutionary analysis. Therefore, a cluster network is more a data-display network than an evolutionary.

4.2 Galled trees

Before we come the the next two networks, I want to introduce the concept of *galled trees*. A galled tree can be seen as an intermediate step between a tree and a network. If we are precise, it actually is a kind of network and no tree. Huson, Rupp, and Scornavacca (2010, p. 156) defines a galled tree as follows:

Definition 4.2 *A rooted phylogenetic network N is called a galled tree, if every non-trivial biconnected component of N properly contains exactly one reticulation. Galled trees are sometimes assumed to be bicombining.*

A biconnected component is a component in the tree which is connected twice. This means if one (reticular) edge is removed, the tree node is still connected within the tree. The becombining condition only appropriate if a galled tree is constructed out of two trees. If it is reconstructed using only clusters, the condition can be dropped (Huson, Rupp, & Scornavacca, 2010). Given the set of clusters C, we already used above: C = $\{\{a\},\{b\},\{c\},\{a,b\},\{b,c\}\}$. There can be two trees, from which the set of clusters could result:
The corresponding network representing this incompatible set of clusters would be a galled tree.
The node b is connected with a as well as with c. This is the expected result for the representation of the set of clusters. The network only has one reticular node, therefore it is a galled tree.
The galled tree is a preliminary state to the other rooted phylogenetic networks. The theory and the algorithms expect that the networks have more than one reticulation node and are therefore more general than a galled tree. Therefore, a galled tree is mostly not used for representing evolutionary events, because it could only represent one. It is too much restricted for a more explicit representation of the events. If we assume a set of clusters which include more than one incompatible clusters, a galled tree would not be able to represent them all. Therefore, a more general network is needed.

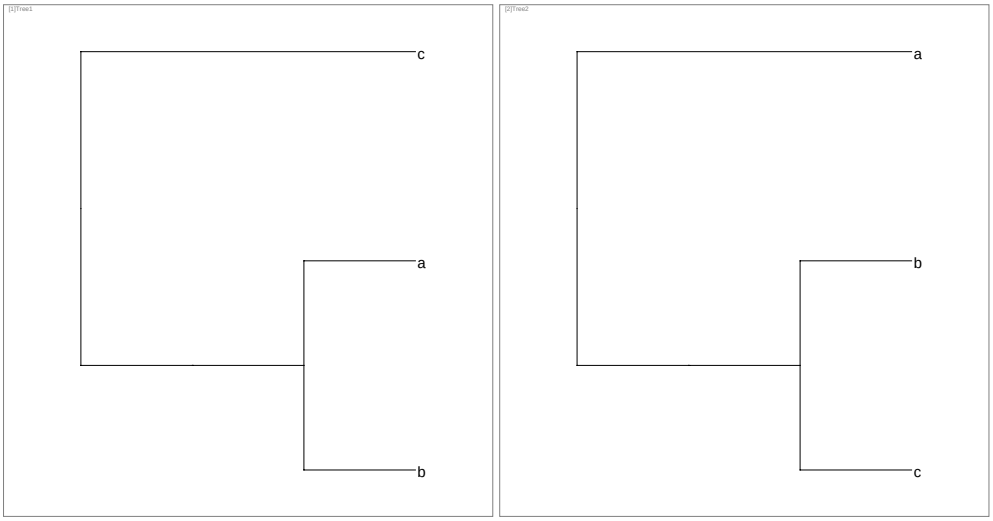


Figure 4.3: A representation of two trees

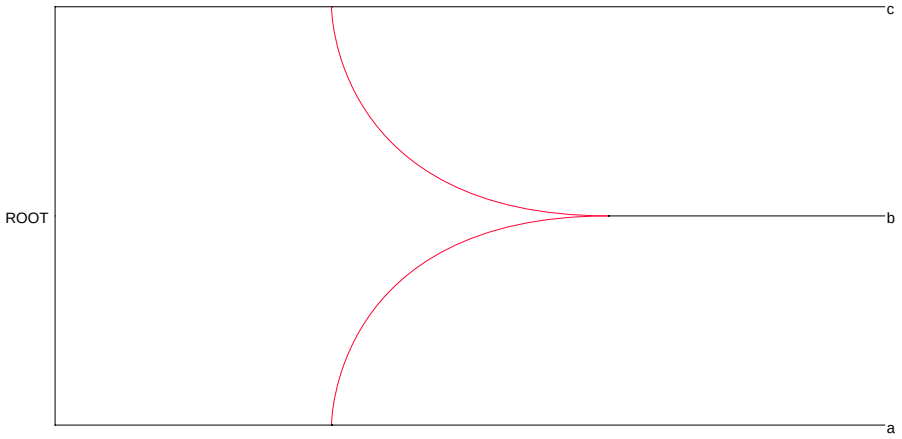


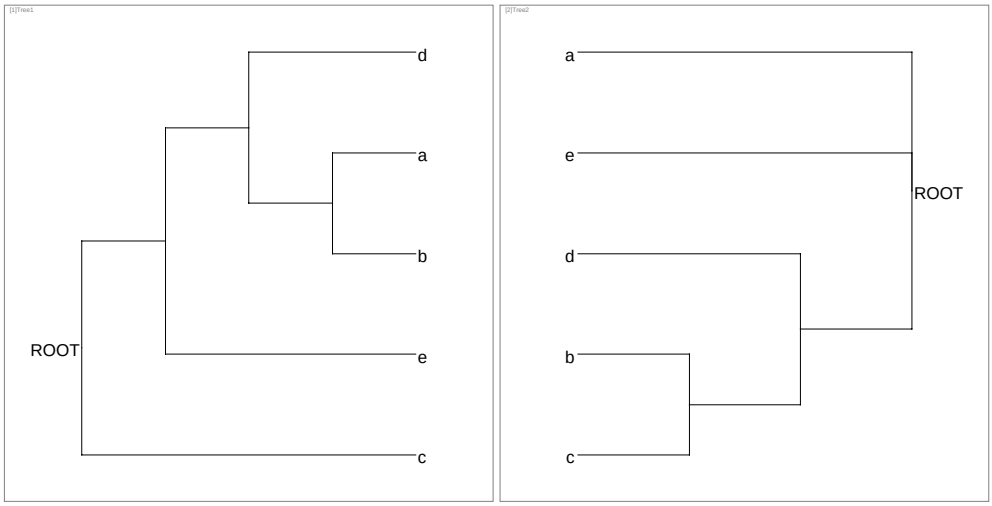
Figure 4.4: A representation of a galled tree

4.3 Level-k Networks

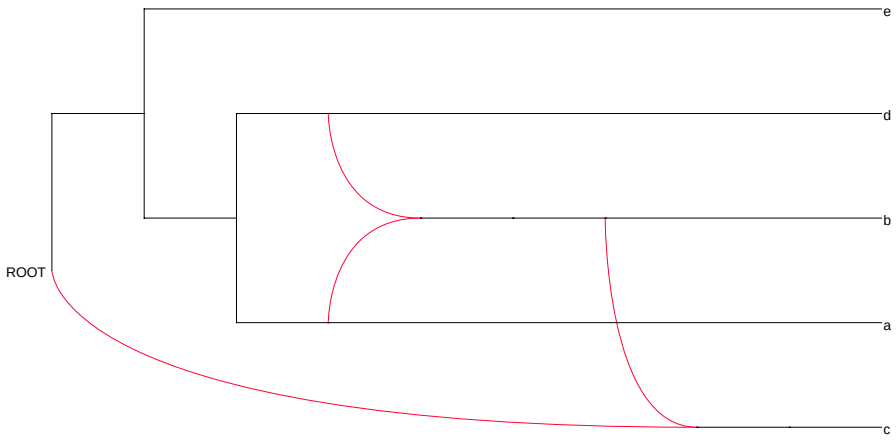
A *level-k network* is a rooted phylogenetic network. K is a variable for the "maximum number of biconnected reticulation nodes within a biconnected component of the network" (Morrison, 2011, p. 123). A level-k network is computed out of two or more trees. The trees are compared and a set of clusters is created. Using this set of clusters, the network is calculated and created. Huson, Rupp, and Scornavacca (2010, p. 160) defines a level-k network as follows:

Definition 4.3 *Let N be a bicombining, rooted phylogenetic network on χ. If the number of reticulations properly contained in any biconnected component of N is k, then N is called a level-k network.*

Due to the definition, a level-0 network is a tree and a level-1 network is a galled tree (Huson, Rupp, & Scornavacca, 2010). Therefore, a level-k network is a generalisation of a galled tree, because it can represent more than one reticulation node on different levels.
We can use the same trees as we used for computing a cluster network and can compute a level-k network.



(a) Two trees



(b) A level-k network

Figure 4.5: A representation from two trees to a level-k network

In figure 4.5(b) the network is a level-2 network represented by two reticulation nodes on different levels.
The pseudo code of the algorithm implemented in Dendroscope is given in Huson, Rupp, and Scornavacca (2010) on page 212. Basically, the algorithm is given a set of clusters. If this set is k=0, the algorithm returns the corresponding tree. Otherwise a taxon is removed and the smaller set of clusters is collapsed. During

collapsing the smaller set of clusters, more non-trivial clusters are removed until the algorithm can compute a tree. The removed clusters are added and a network is built recursively.
As already stated above, the input of the algorithm could also be a set of trees. In our case, this would be a language tree and a concept tree. The level-k network represents the matches and mismatches of both trees according to the algorithm. The clustering within this network is similar to the clustering in the concept tree.

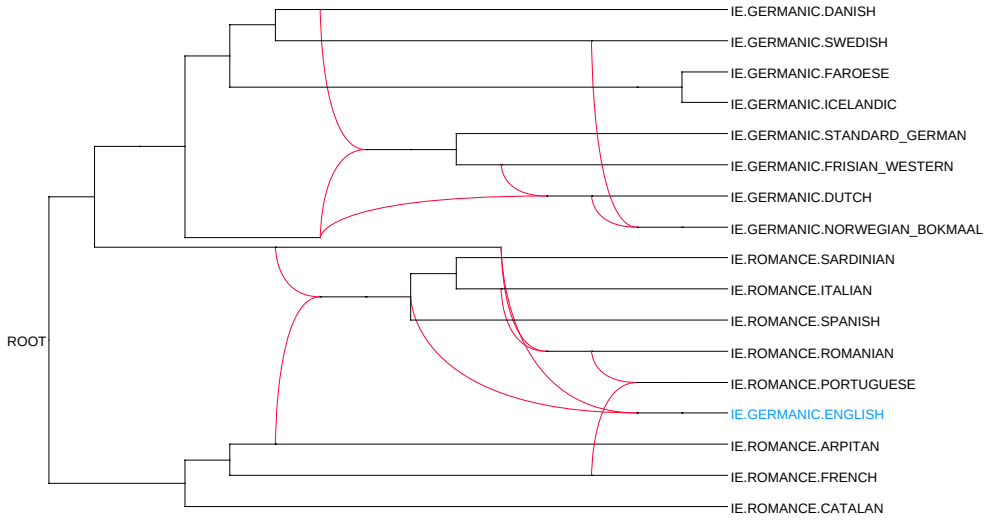


Figure 4.6: A level-k network of a language tree and a concept tree

This is the case, because English is grouped together with the Romance languages and not with the Germanic languages. English is still connected to the Germanic languages due to a reticulation edge and it is connected to Romanian, which has a similar word for *mountain* than English. The other reticular edges (red) can either indicate other evolutionary events or insufficient data. The latter can also be the case, because it is well known that linguistic data is not that sufficient than most of the biological data. Nevertheless, a level-k network is able to model evolutionary events and could therefore be a good candidate for further linguistic analysis (Morrison, 2011).

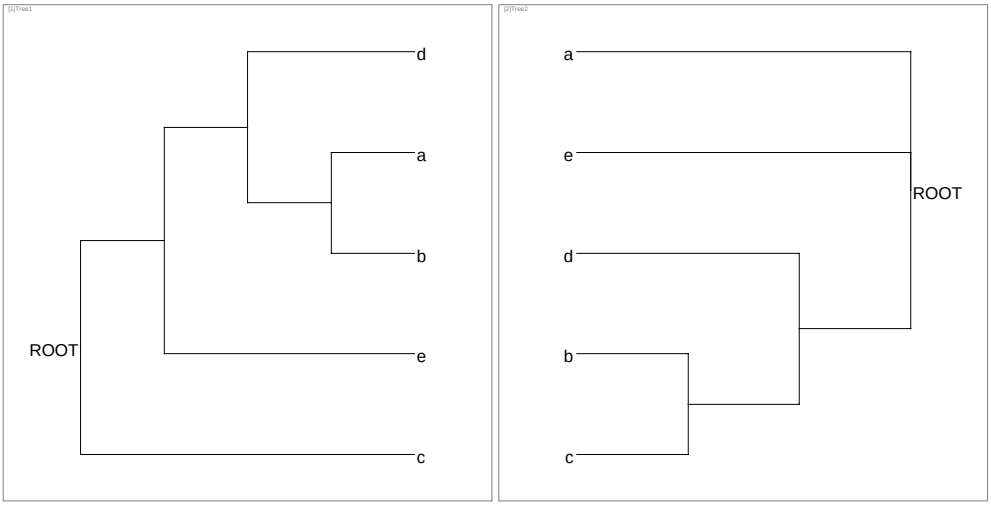
4.4 Galled Networks

A *galled network* is also a rooted phylogenetic network. It is a generalized version of the galled tree, because it can have more than one reticular edge and has therefore less restrictions to be computed than a galled tree. It is said to be a more general type of network, but not so general than a level-k network (Morrison, 2011). Additionally, a galled network is able to represent every representation of a given set of clusters. This might be not the case for a galled tree or even for

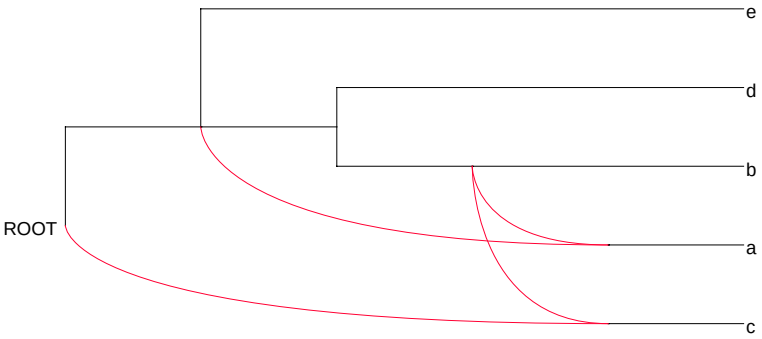
level-k networks. Huson, Rupp, and Scornavacca (2010, p. 163) defines a galled network as follows:

Definition 4.4 *A bicombining rooted phylogenetic network N is called a galled network, if every reticulation in N has a tree cycle.*

A tree cycle is similar to a reticulation cycle. According to Morrison (2011, p. 210), a reticulation cycle is "a set of arcs in a directed acyclic graph that form a cycle or a loop if the arc directions are ignored". A reticulation cycle is also called a *gall* (Morrison, 2011).
For computing a galled network, two or more different trees are needed. A set of clusters is created out of the given trees and an algorithm is used for computing the network. Every set of clusters can be represented by a galled network, therefore there exists a galled network for any set of clusters (Huson, Rupp, & Scornavacca, 2010).
We can use the same trees than above to compute a galled network.
In figure 4.7(b), the galled network includes two reticulation nodes. The representation differs from the one given in figure 4.5(b). This is due to the different algorithms. The pseudo code of the algorithm to compute a galled network is given in Huson, Rupp, and Scornavacca (2010) on the pages 204-205. The algorithm has two different stages.
In the first stage, a minimum set of reticulations is determined. The algorithm builds a set of so called *reticulate taxa*. This set contains incompatible taxa, which is responsible for reticulations within the network. This taxa is removed from the whole taxon set. The remaining set is called *maximal compatible subset* (Huson, Rupp, & Scornavacca, 2010). The clusters are now compatible and can be represented by an underlying rooted tree (Huson, Rupp, & Scornavacca, 2010). Two cluster sets remain: C with all compatible clusters and C' with all incompatible clusters.
In the second stage, the remaining incompatible clusters need to be attached to the underlying tree in the best possible way (Huson, Rupp, & Scornavacca, 2010). This is called the *minimum attachment problem*. By solving this problem, a minimum galled network is computed.
This algorithm can also be used for computing a galled network on a language tree and a concept tree. The galled network can also represent the matches and mismatches between the trees.
Within this network, the clustering is similar to the one of the language tree. English is represented in the middle of the tree and is connected to the Germanic languages and also to the Romance languages. The galled network can represent this mismatch in an appropriate way in which English is neither part of the one



(a) Two trees



(b) A galled network

Figure 4.7: A representation from two trees to a galled network

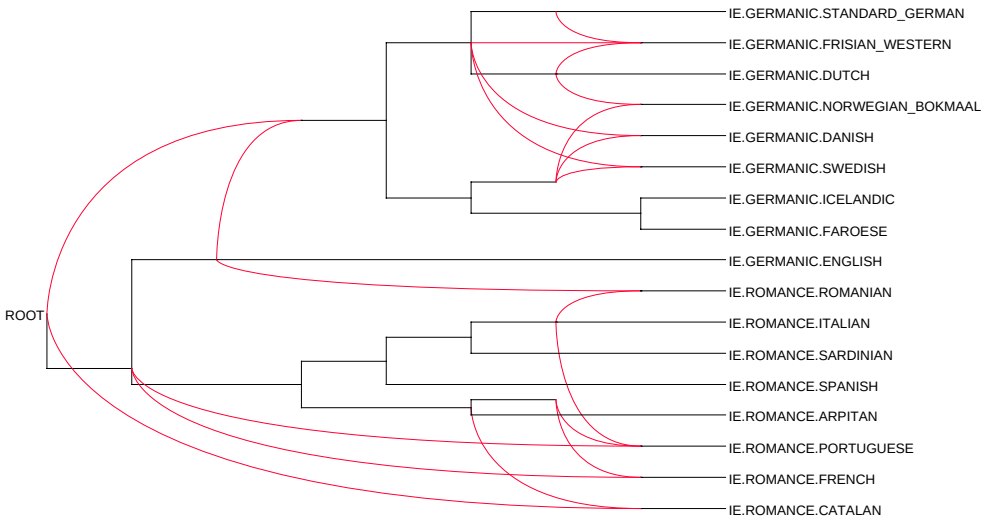


Figure 4.8: A galled network of a language tree and a concept tree

language family nor of the other. It is ordered in the middle more as a kind of connector between the language families. The other mismatches could either indicate more evolutionary events or insufficient data, similar to the ones represented in the level-k network. I already stated above, that insufficient data is nothing new within linguistics and it is well know that biological data is more sufficient than linguistic data. It would therefore not be surprising if some mismatches are due to insufficient data.

4.5 Comparison of the tree Algorithms

The three algorithms explained above all have advantages and disadvantages over the others. The question is, which one is the best algorithms to construct a network that can represent linguistic data. In our case, other things need to be taken into account than for biological data.
The first difference is the hardwired and softwired interpretation. Huson, Rupp, and Scornavacca (2010, p. 147) defines them as follows:

Definition 4.5 *Let N be a rooted phylogenetic network on χ. Under a hardwired interpretation, any edge e in N represents precisely one cluster $\gamma(e)$, namely the set of all taxa that label nodes that are descendants of the target node of e. Under a softwired interpretation, any tree edge e in N represents a set C(e) of one or more clusters. Each member of this set is determined as follows: First, for each reticulation r, turn one in-edge on and all others off. Then, for each such set of choices, a cluster belonging to C(e) is given by the set of all taxa that label nodes that lie below e and can be reached without using and reticulate edge that is off.*

According to Huson, Rupp, and Scornavacca (2010), a softwired network will require fewer edges to represent the clusters given by the two input trees, than a hardwired network. Within a hardwired network any edge only represents one cluster and is therefore more precise in representing mismatches. Therefore, hardwired networks are more specific and should be used for out purpose.
All networks represented above are rooted phylogenetic networks and the methods to compute them are all cluster-based (Huson, Rupp, & Scornavacca, 2010; Morrison, 2011). The rooted phylogenetic networks are needed to represent a set of incompatible clusters and these incompatible set of clusters are able to represent evolutionary events or insufficient data. All three algorithms have this in common and are able to represent the matches and mismatches of trees in their corresponding network.

The cluster network is a hardwired and abstract network (Morrison, 2011). It needs more reticulation nodes to represent the mismatches within the network

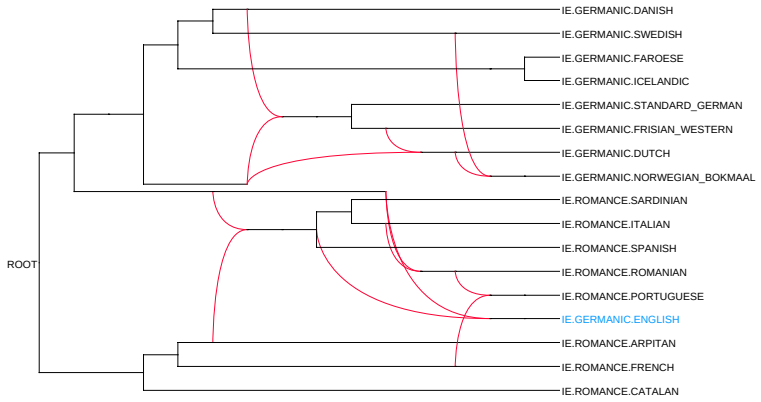
than the others. Additionally, it is an abstract network. This means, that the networks is not able to illustrate evolutionary events and is mostly used to display the data. Morrison (2011) already stated that a cluster network is not able to model evolution and is therefore separated from the other networks. In other words, not all reticulation nodes within the cluster network shown in figure 4.2 represent evolutionary events. All in all, a cluster network represents a collection of rooted trees according to their mismatches. This is the reason why Morrison (2011) separates cluster networks form other networks.
A cluster network can be seen as an alternative to a *consensus tree*. A consensus tree is a species tree created out of two or more gene trees, if there is no species tree present. A cluster network represents the clusters present in the comparing trees and the arcs represent the cluster which is formed by all taxa descending from that arc. An arc or a reticular edge within this network might not only represent evolutionary events but only clusters.
This leads us to the result, that this algorithm might not be the right one for our purpose. The combination of trees or the mapping of a concept tree into a language tree should indicate evolutionary events which might have caused the mismatches and it should not only display the clustering of the trees. Additionally, it is well known that linguistic data might be more insufficient than biological data. This means, that some mismatches are due to the data and not to evolutionary events. The more mismatches we have, the more difficult is it to differentiate between evolutionary events and mismatches due to the data.

The level-k network and the galled network are both hardwired and evolutionary networks. Both need less reticulations to represent the mismatches than the cluster network. The networks are able to model evolution and the mismatches can indicate evolutionary events. Additionally, the insufficient data plays also a role. The reticulation nodes can either be evolutionary events or due to insufficient data. Nevertheless, there are less nodes to look for than within a cluster network. We also need to keep in mind, that a galled network is also a level-2 network, whereas a level-2 network is not a galled network. This makes the level-k network more general than the galled (Morrison, 2011).
Huson, Rupp, and Scornavacca (2010) compared the algorithms of the galled network and the level-k network.

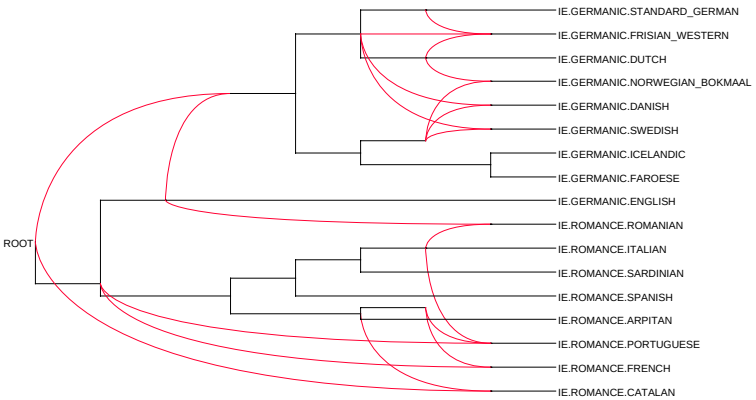
galled network	**level-k network**
runs faster	runs slower
more reticulations	less reticulations

Table 4.1: Differences between the algorithm of galled networks and level-k networks

He comes to the solution that the algorithm for the galled network runs faster than the other, but the algorithm for the level-k network creates a network with fewer reticulation nodes. In phylogenetics, the level-k network is on his way to become the more general tool to compute a network form different types of data (Huson, Rupp, & Scornavacca, 2010).
According to this comparison, one might think that the level-k network has the better algorithm because it computes less reticular nodes. This might be right for biological data, but what about linguistic data? If we compare the level-k network and the galled network for linguistic data, do we come to the same solution?



(a) A level-k network



(b) A galled network

Figure 4.9: A comparison of a level-k network and a galled network

The galled network and the level-k network differ in the number of reticular nodes. The galled network has two reticulations more than the level-k network. This is as expected from phylogenetics. The more problematic difference is the clustering. Within the level-k network in figure 4.9(a), English is clustered between the Romance language and is only connected with one reticular edge to the Germanic. This is the case, because within the concept tree English is grouped together with the Romance language. English borrowed the word *mountain* from old French and within the concept tree, English is more a Romance language than a Germanic.

This should not be the case for the network. English belong to the Germanic language family and should be grouped as such. This is the case within the galled network. English is grouped between the Germanic languages and the Romance languages and is connected to both with a reticulation edge. In our case this is the more adequate solution. Additionally, this is only a sample set of language. If we want to compare trees of all Indo-European languages or even more language families, we need a fast algorithm. I tested the computation of a level-k network and of a galled network, while constructing a network with the Indo-European trees. The computation of a galled network is faster than the construction of a level-k network. Therefore, the algorithm of a galled network might be better to compare linguistic trees than the other algorithms.

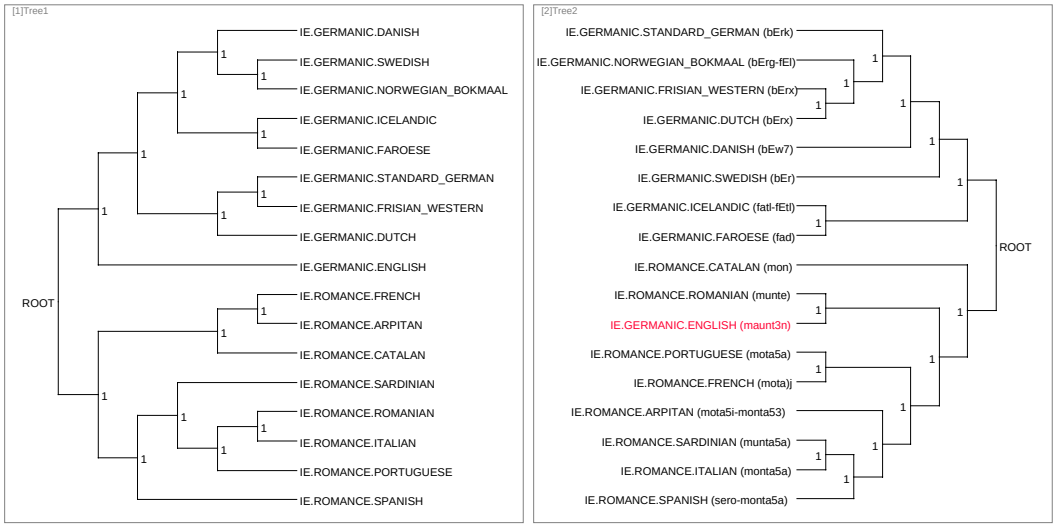
4.6 Evaluation of the Network

There are two different ways of constructing a galled network. First, the computation of a network out of more gene trees or in our case out of more concept trees. Second, the computation of a galled network due to a comparison of a language tree and a concept tree.

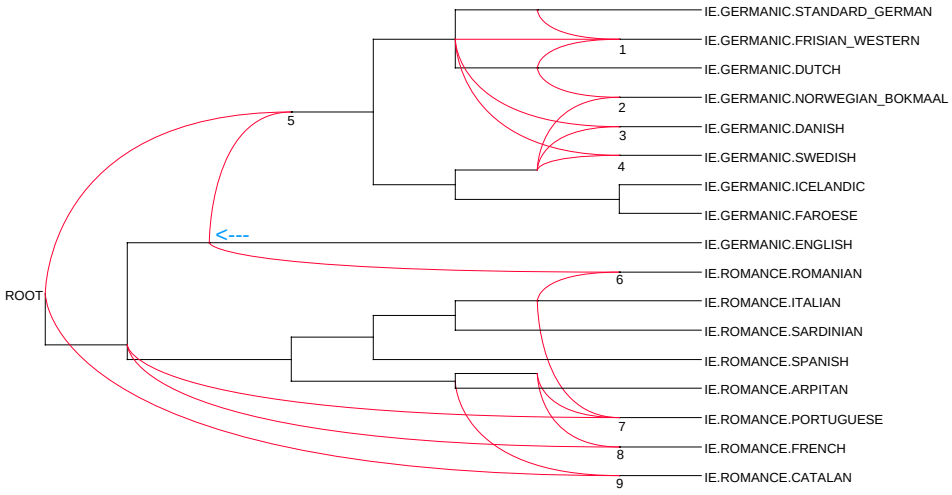
The idea of constructing a network out of 40 concept trees comes from the biological point of view. As already stated, within phylogenetics networks are mostly created out of two or more gene trees (Huson, Rupp, & Scornavacca, 2010; Morrison, 2011). This is a popular method to create a kind of species tree out of gene trees, if the species tree cannot be computed in another way. In linguistics, mostly a species tree can be computed out of the given data, as it can be seen in the previous section. This is the case, because within linguistics a small dataset can already be useful for creating an appropriate language tree (Brown et al., 2008). Therefore, the method to create a network which serves as a species tree is not needed.

The second idea of matching a concept tree to its corresponding language tree comes also from phylogenetics. Within phylogenetics, gene trees can be mapped to their species trees to reconstruct evolutionary events (Huson, Rupp, & Scornavacca, 2010; Morrison, 2011). Nevertheless, methods for this mapping are under their process of invention. In section 2.3 Atkinson and Gray (2005) stated that for example the counterpart for borrowing is horizontal gene transfer. It might be easy to adapt algorithms for horizontal gene transfer into linguistics, but there is no popular algorithm which can be used from bioinformatics. There are a lot of theoretical papers on horizontal gene transfer, but less implemented

algorithms. This is one reason to use Dendroscope on linguistic data. It might be the case that already implemented methods in phylogenetic work on linguistic data. This is the case for algorithms to compute a distance matrix and a phylogenetic tree. But what about using the algorithm of a galled network to combine a concept tree and a language tree? Does it gives us good and expected results?



(a) Concept tree "mountain" and language tree



(b) The galled network

Figure 4.10: Two linguistic trees and their corresponding galled network

Figure 4.10(a) shows the language tree and the concept tree with the corresponding representation of the concept "mountain" and figure 4.10(b) represents the galled network computed out of the two trees. The reticular nodes are labelled with numbers. Within this networks, the reticular edges represent evolutionary events or are due to insufficient data. I already stated that insufficient data is nothing new within linguistics. Nevertheless, before I claim that some reticulations are due to insufficient data, I want to explain the reticulations which are

evolutionary events.
The representations of the concept "mountain" are clustered according to distances between the concepts. The smaller the distance, the closer are the languages grouped together. Therefore, a new clustering may emerge. This is the case for each reticular node within the galled network shown in figure 4.10(b). Most of the reticulation are due to relationships between languages within the same language family. Within a language family, different changes can appear.

1. Sound change
2. Morphological change
3. Borrowing

Sometimes the phonetic form of a word changes and becomes more similar to the one of another language but can still be recognized as a word of the same language family. This phenomena is known as *sound change* (Campbell, 2013). Another phenomenon is the morphological or lexical change. The form of a word changes lexically and is taken over into the language. This change leads to different representations of the word (Campbell, 2013). However, we also need to keep in mind that borrowing within a language family can be possible. There are some words which are borrowed from Middle Low German (spoken in the North of Germany and the Netherlands) into the Skandinavian languages between 1300 and 1550 AD (Schülke, 2005). All three phenomena can take place within a language family. The words can still be recognized as a word of the same language family, although small changes might have taken place. We need to keep in mind, that the algorithm is not able to differentiate between these three things. If the words within a language change, due to one of the listed phenomena, the identification of the events need to be made by interpretations and predictions.
Additionally, evolutionary events between two language can take place. These events are mostly due to language contact. If two languages are in contact with each other, the most frequently event which takes place is the borrowing of a word (Campbell, 2013). Within the process of borrowing, the word is adapted into another language and the above listed phenomena, like sound change and morphological change, can also take place. The events are only separated by events within a language family and events within two languages of a different family. The only evolutionary event which takes place between two unrelated languages, is indicated with the blue array in 4.10(b). English borrows the representation for the concept "mountain" from the Romance language family. More specifically English borrowed the word from Old French *montaigne* as it can be seen in section 2. Therefore, these reticulation edges are the only ones which

represent a well-known evolutionary event between two language families.
According to Morrison (2011), there is no suitable method to reconstruct an evolutionary network within phylogenetics. Why should there be a suitable one for linguistics? The idea ofMorrison (2011) for an evolutionary network is a phylogenetic tree with some reticulations. The galled network is an evolutionary network which is able to represent evolutionary events. As we saw above, the reticulations are only identifiable due to predictions, interpretations or background knowledge. Morrison (2011) states that a network should be simple, should not contradict with existing opinions and should additionally allow realistic predictions. For the creation of a suitable method for the reconstruction, three basic things need to be taken into account (Morrison, 2011):

(i) Randomness

(ii) Reticulations

(iii) Rooting

Randomness plays a crucial role during the computation of the phylogenetic tree. Most tree-building methods address the issue of stochastic variation (Morrison, 2011). Mostly, the variation is present within trees with branch lengths. This is also the case for our trees. However, the trees are calculated with a sufficient algorithm. We use the dERC algorithm, stated in (Jäger, 2013), to compute the distance matrices and the FastME algorithm to reconstruct the phylogenetic tree. Both algorithms are explained in section 3. The resulting language tree, for all languages present in the ASJP database, is similar to language trees created by experts (Jäger, 2013). Therefore, randomness does not play such a crucial role within the computation of the trees but may play a role within the computation of a network.
Reticulations should be present within each network. They represent the evolutionary events which might have happened during history (Morrison, 2011). Morrison (2011) differentiate between reticulations which any network should be able to quantify (hybridization, horizontal transfer, recombination) and reticulations which can also be represented by other than networks (duplication-loss, deep coalescence).
Rooting is the most important thing which need to be taken int account. For Morrison (2011), rooting is a problem which need to be solved or at least need to be dealt with. Differences between trees could also be due to rooting not only to evolution. The rooting affects the rest of the tree, depending on the outgroup or random rooting. The rooting might affect the whole clustering, whereas randomness might only affect the typology of the tree (Morrison, 2011).

5 Comparing phylogenetic trees using Distances

As we saw in the preceding section, there is a lot to investigate in the processing of evolutionary networks. According to Morrison (2011, p. 158) we need to be aware of three basic things which he states as follows:

Approach 5.1 *We first need a consistent root for all of the [...] trees, and little (or no) conflict caused by stochastic variation. In essence, we need a two- or three step strategy for producing evolutionary networks in these circumstances: networks on their own do not distinguish vertical from horizontal inheritance*

This idea concentrates on the visualisation of networks and not on their computation or other underlying algorithms. Before addressing the visualisation of a network, I want to state my idea of the detection of evolutionary events with distances.

Instead of using a clustering algorithm, the idea is to detect evolutionary events by measuring the distance between trees. Trees can be compared by calculating their distance. A small distance indicates a small difference or a greater similarity between the trees (Huson, Rupp, & Scornavacca, 2010). Each inner node in a tree indicates a *split* of one language into two. Two trees are identical, if they share the same splits and therefore have the same tree structure. To detect evolutionary events, it is not sufficient to compare just two trees of the same language set. A single distance measure cannot locate the source of the difference between the trees. The idea is to remove one language at a time from the language and the concept tree. These trees are compared and their distance is measured. If the distance gets smaller, the trees became more similar. Therefore, the missing language has a great impact on the original structure of the trees. Thus, this language is partly responsible for the difference between the trees and might have caused an evolutionary event. This is done for each language in the language sample. This idea was implemented using different distance measures. These distances are explained and compared below.
For each concept, there are as many distances as languages in the sample. The next step is to discover the languages causing a significant distance between trees. A threshold is needed to find those languages. The removal of one language at a time changes the distances between the trees, even if not significantly. Only those languages which are under a specific threshold might be responsible for causing an evolutionary event. Languages which are responsible for evolutionary events are expected to differ significantly from the mean of the observed distances. Distances differing more then 5% from the mean are said to be significant. Therefore

the threshold is located at 95% of the mean. All languages below this threshold are considered to be outliers. With this measurement, we can avoid that small differences between trees automatically cause an evolutionary event.
In the next step, a network is computed. All languages causing evolutionary events are coloured. Their connections within the language and concept tree are visualized using reticulations. The original language tree functions as the underlying tree for the network. The languages causing events have one connection to their parent node in the language tree and one connection to their sister node in the concept tree. These connections are the reticulations which should indicate the evolutionary events. The pseudocode for the computation of the reticulations is given in appendix G.
With this algorithm, we cannot distinguish between different evolutionary events during language history. But in most cases, we can minimize the number of reticulations compared to the pyhlogenetic networks.
Coming back to the idea of Morrison (2011), the visualization of the network is clearly arranged and the reticulations are limited. Nevertheless, the reticulation only indicate mismatches between trees caused by some languages. The network cannot depict different types of evolutionary events. This can only be done through interpretations, predictions and background knowledge. To ensure a consistent root, the user can specify an outgroup. Both trees are rooted according to the defined language(s).[8]

5.1 Distances

Distance measures can be used to compare phylogenetic trees. Within phylogenetics, these measurements are used to compare trees which are constructed using different methods (D. Robinson & Foulds, 1981). Different construction methods could lead to different results. Therefore, it might be the case that two trees with the same taxa set are constructed differently. To measure this difference, the distances between the trees are computed. The higher the measurement, the more different are the trees.
This method can also be used to compare a species tree and a gene tree under the assumption that both trees share the same taxa set. This is the only precondition for the estimation of the distance between two trees. Therefore, it does not matter if the trees are computed by different methods or if one of the trees is a language and one a concept tree.
Within linguistics, we can benefit from the distance measures used in bioinformatics. We can compute one language tree and one concept tree. To compare

[8] The program can be provided by the author.

them, we can simply compute the distances between them. Phylogenetics provide different distance measurements. The question is, which one is the best for the comparison of a language and a concept tree. Additionally, we want to detect evolutionary events by leaving out one language at a time and computing the distance between the trees. To answer this question, three different distance measurements are tested and compared. The best is used for further analysis and the reconstruction of the network.

5.1.1 Robinson-Foulds Distance

The Robinson-Foulds distance is one of the most popular distance measures within phylogenetics. D. Robinson and Foulds (1981) introduced their theory in 1981. The main idea was to present a method for combining any pair of trees. This was the first approach to calculate the distance measures for binary and non-binary trees.
There are two kinds of approaches for measuring the distance between two trees. One is to ignore the weights of the branches and compare only the topology of the tree structure. The second one is to take the weights into account "where each line has a weight equal to the number of mutations between the sequences it connects " (D. Robinson & Foulds, 1981, p. 132). The first approach was taken by D. F. Robinson (1971) and several other authors. The second approach was used by D. Robinson and Foulds (1979). A further step, introduced by D. Robinson and Foulds (1981) was to create a metric "on the set of all phylogenetic trees labeled with n species" (D. Robinson & Foulds, 1981, p. 132) and this metric defines the distance between the trees.
The original algorithm introduced by D. Robinson and Foulds (1981) was used to measure the operations of the transformation from one tree into another. Therefore, the number of edge contractions and decontractions are computed. These contractions and decontractions are required to transform one tree into another. The idea of contractions is to form a new tree by removing an edge and form a union of the nodes containing both labels which are connected by the edge (D. Robinson & Foulds, 1981). The idea of a decontraction is the other way around. A new tree is formed be inserting a new edge and split the node into two. The labels are splited accordingly. If the node only has one label, this label is associated with one new node and the other new node gets an empty label. The contractions and decontractions are used for computing the distance metric. Therefore, D. Robinson and Foulds (1981) define the distance in the following way:

Definition 5.2 *For each positive integer n we define d_n to be the maximum dis-*

tance between two phylogenetic trees on n species.

Since the edge can be seen as a split, the contraction can be seen as a removal of a split and the decontraction can be seen as an addition of a split (Huson, Rupp, & Scornavacca, 2010). Taken this into account, Huson, Rupp, and Scornavacca (2010, p. 61) defines the Robinson-Foulds distance in the following way:

Definition 5.3 *The Robinson-Foulds distance between two unrooted phylogenetic trees T_1 and T_2 on χ is based on the symmetric difference of the sets of splits represented by two trees:*
$d_RF(T_1, T_2) = \frac{1}{2}|S(T_1) \triangle S(T_2)|$

The symmetric difference between two trees is known as the distance between two trees which is based on the number of branch lengths that differ between the two trees. It is also known as the partition metric, which was introduced by D. Robinson and Foulds (1981). The length of the branches is ignored so the tree can be seen as a set of branches (Felsenstein, 2004). "Each branch divides the species into partitions with two sets, one connected to each end of the branch" (Felsenstein, 2004, p. 529). For each tree, such partitions are defined and the difference is computed by counting all partitions of one tree which are not shared within the other tree (Felsenstein, 2004). Felsenstein (2004, p. 529) states that "the symmetric difference is easy to compute, but it is highly sensitive to all differences between the trees".
Originally, the Robinson-Foulds distance was intended to compute the distance between two unrooted trees. Nevertheless, it distance can also be used to compute the distance between two rooted trees. The branches also define a set of species, but they are connected to its upper end. "The symmetric difference is then the number of sets that differ between the two trees" (Felsenstein, 2004, p. 530).

5.1.2 Quartet Distance

The quartet distance is another popular method to measure the distance between two trees. The main idea of this approach goes back to Estabrook, McMorris, and Meacham (1985). They present their algorithm which splits the trees into quartets and computes the distance between the trees according to the quartets which differ between them. To get a better idea of the algorithm, Huson, Rupp, and Scornavacca (2010) discuss the restriction of a phylogenetic tree into a subset of its taxa. They give the following definition:

Definition 5.4 *Let T be a phylogenetic tree on χ and let $C \subset \chi$ be a subset of taxa. We use T(C) to denote the minimum connected subgraph of T that contains*

all leaves that are labeled by the elements of C. The restriction of T to C is defined as the phylogenetic tree $T|_C$ that is obtained from T(C) be suppressing all suppressible nodes.

Within the quartet distance the subset C contains four leaves of T. The set contains the labels and is therefore called a *quartet tree* (Huson, Rupp, & Scornavacca, 2010). According to the quartets which can be created out of a tree, a set containing all quartets can be defined. Out of this definition, Huson, Rupp, and Scornavacca (2010, p. 62) states the definition of the quartet distance.

Definition 5.5 *The quartet distance between two unrooted phylogenetic trees T_1 and T_2 on χ is defined as*
$d_{quartet}(T_1, T_2) = \frac{1}{2}|\mathcal{Q}(T_1) \triangle \mathcal{Q}(T_2)|$

This distance measurement is "more sensitive to partial similarities of structure between trees" (Felsenstein, 2004, p. 530). The approach introduced by Estabrook et al. (1985) seems difficult to compute, this is the case because "the number of quartets is proportional to the fourth power of the number of species" (Felsenstein, 2004, p. 530). Nevertheless, Brodal, Fagerberg, and Pedersen (2004) discovered an algorithm which can compute the quartet distance in time $O(n \log n)$. This algorithm is used to compute the distance between two unrooted phylogenetic trees. Brodal et al. (2004) explain the underlying quartet distance of the algorithm in the following way:

Definition 5.6 *Given two evolutionary trees T_1 and T_2 on the same set S of species, the quartet distance between the two trees is the number of four-sets $a, b, c, d \subseteq S$, for which the quartet topologies in T_1 and T_2 differ. As there are $\binom{n}{4}$ different four-sets in S, the quartet distance can also be calculated as $\left(\binom{n}{4}\right)$ minus the number of four-sets for which the quartet topologies in T_1 and T_2 are identical.*

The algorithm is implemented in the program *qdist* by Mailund and Pedersen (2004). It provides different calculations and their results. One of them is the normalized quartet distance. It is calculated by dividing the quartet distance by the number of possible quartets over their leaves.[9] The normalized quartet distance is used further within this paper. With the algorithm, the quartet distance between two trees can be computed efficiently and rapid. Therefore, the algorithm provides an attractive and comfortable way that enables the user to benefit from the computation of the quartet distance.

[9]This number is calculated through the binominal, i.e. for n leaves the number of possible quartets is computed by $\binom{n}{4}$.

5.1.3 Triplet Distance

The triplet distance is another distance measurement between two trees. The first algorithm to compute the triplet distance was introduced by Critchlow, Pearl, and Qian (1996) in 1996. They gave a formal definition of the triplet distance in their article.

Definition 5.7 $S_n = \sum_{ijk} I_{ijk'}$

A triplet is a set which consists out of three leaf labels (Brodal, Fagerberg, Mailund, Pedersen, & Sand, 2013). "This is the smallest number of leaves for which the subtree induced by these leaves can have different topology in two rooted trees T_1 and T_2" (Brodal et al., 2013, p. 1815). The underlying idea of the triplet algorithm is the same than for the quartet distance. It splits the trees into triplets and computes the distance between them according to the triplets in which they differ. The restriction of a phylogenetic tree into a subset of its taxa, as already stated in definition 5.4, remains the same.
Huson, Rupp, and Scornavacca (2010, p. 62) defines the triplet distance in the following way:

Definition 5.8 *The rooted triplet distance between two rooted phylogenetic trees T_1 and T_2 on χ is defined as*
$d_{triplet}(T_1, T_2) = \frac{1}{2}|\mathcal{R}(T_1) \triangle \mathcal{R}(T_2)|$

In contrast to the quartet distance, each triplet is anchored at a node. "For triplets, this anchor node is the lowest common ancestor in [the tree] of three leaves" (Brodal et al., 2013, p. 1861). "[The triplet distance] can naïvely be computed by enumerating all $(\binom{n}{3})$ sets of three [leaves] and for each comparing the induces topologies in the two trees" (Brodal et al., 2013, p. 1815).
Further improvements of the algorithm were made by Sand, Brodal, Fagerberg, Pedersen, and Mailund (2013) and Brodal et al. (2013) in 2013. Brodal et al. (2013) introduced an advanced algorithm to compute the triplet distance between trees of arbitrary degrees. This includes also binary trees. An overall idea of the algorithm is stated within their article (Brodal et al., 2013, p. 1816).

> For a node $v \in T$ we denote by τ_v the set of triplets anchored at v. Then $\{\tau_v | v \in T\}$ is a partition of the set $\mathcal{T}$ of triplets. Thus, $\{\tau_v \cap \tau_u | v \in T_1, u \in T_2\}$ is also a partition of $\mathcal{T}$. Our algorithm will find $A(T) = \sum_{v \in T_1} \sum_{u \in T_2} A(\tau_v \cap \tau_u)$, where $A(S)$ on a set S of triplets is the number of triplets being resolved in both T_1 ans T_2, and having the same topology in both trees.

The algorithm can compute the triplet distance between two rooted trees in time $O(n \log n)$. A program containing the triplet distance is provided by Sand et al. (2014). The program calculates the distance in the above mentioned way. To compute the normalized triplet distance, we need to divide the distance by the the number of possible triplets over their leaves.[10] The normalized version is used further in this paper. With the algorithm, the triplet distance can be computed in an efficient way. In contrast to the quartet distance, the triplet distance requires two rooted trees instead of two unrooted. Therefore, the triplet distance provides an additional feature which enables the user to benefit from the algorithm.

5.1.4 Comparison of the distance methods

For the comparison of the three distance measures, each algorithm is applied to the same data set. The data set contains the same language sample of Germanic and Romance languages used in the previous chapters. The corresponding language tree and concept tree of the concept "mountain" are displayed in figure 5.1. These trees contain all languages of the sample and visualizes their relationship to each other.

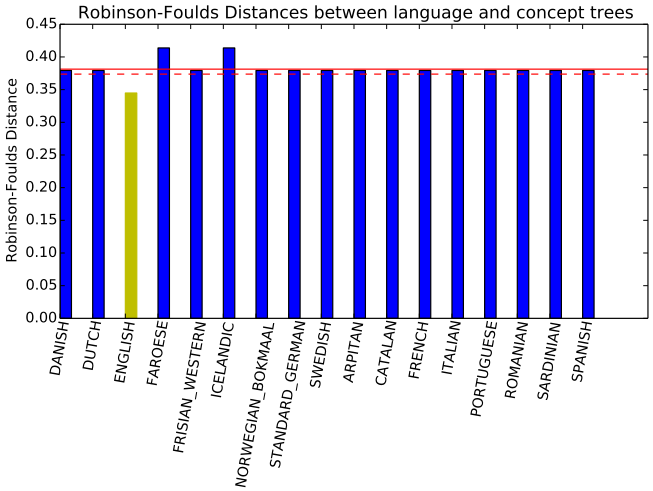
Figure 5.1: The language tree and the concept tree for "mountain"

For each tree, one language is removed and a new tree is created. The trees with the same missing language are compared and the distance is measured. These distances are evaluated within the following bar charts. This is done for each of the three distance measurements. Additionally, a threshold is computed which

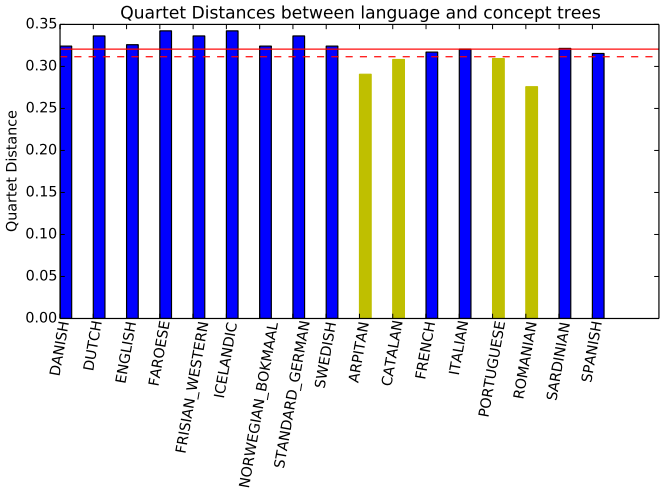
[10] This number is calculated through the binominal, i.e. for n leaves the number of possible quartets is computed by $\binom{n}{3}$.

indicates the outlier languages causing an evolutionary event.

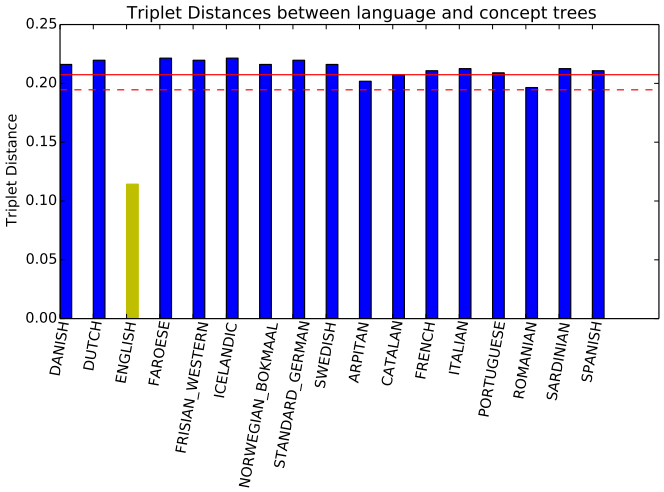
For each distance, a bar chart was created. This bar chart represents the distances between each tree with the same missing language. The name of the distance and the corresponding scale of measurements are placed on the Y-axis. All bars are positioned on the X-axis. The bars represent the distances between the trees with a missing language and are labeled with the corresponding missing language. For each language in the language sample there is a bar. In our case, the language sample contains seventeen languages and therefore there are seventeen bars displayed. The lower endpoint of the threshold is displayed using a dotted red line and the mean by a solid red line. The distances between the trees which causes no statistical significance are coloured in blue and the languages which are under the threshold are coloured yellow.
Each distance has its advantages and disadvantages. On the first sight, the Robinson-Foulds distance and the triplet distance give us the expected results in contrast to the quartet distance. The distance measures are computed for the concept of the tree "mountain". We expect that the trees without English are more similar than the trees including English. This is due to the fact that English is grouped within the Romance languages in the concept tree. Therefore, English should be under the threshold. This can clearly be seen in the bar charts of figure 5.2(a) and 5.2(c). The quartet distance does not lead to the expected result. Instead of English, four Romance languages should cause evolutionary events. The algorithm which computes the quartet distance uses unrooted trees. The quartets are the smallest number of leaves which can be used as an informative subtree (Brodal et al., 2013). The distance between the quartets is used to compute the overall distance between two trees. It might be possible, that the quartets which are formed by the algorithm are formed in such a way, that the absence of English in the tree does not cause a significant difference. Additionally, a big disadvantage of the algorithm is the usage of unrooted trees. The trees displayed in figure 5.1 are both rooted. It can also be seen, that the algorithm does not leads us to the expected results. Therefore, this algorithm is not the best for our purpose.
Two distance measures remain. The Robinson-Foulds distance and the triplet distance. Both measures can be applied to rooted trees. Figure 5.2(a) and 5.2(c) show, that both algorithms lead to the expected result. In both charts, English is coloured as the language which causes an evolutionary event. The decision between the two algorithms is therefore made by their internal advantages and disadvantages.
The Robinson-Foulds distance was at first motivated to compare unrooted trees,



(a) The Robinson-Foulds distance



(b) The quartet distance



(c) The triplet distance

Figure 5.2: The evaluation of the three distance methods for the concept "mountain"

whereas it can also be used to compare rooted trees (Felsenstein, 2004). The algorithm considers all possibilities the leaves and their corresponding labels can be split (D. Robinson & Foulds, 1981). For rooted trees, the algorithm takes only the split of leaves into two sets into account. Informally, this means that all edges or splits which both trees have in common and all different splits or edges are counted. It is not surprising, that the Robinson-Foulds distance is the most frequently used distance measure, because edges are the simplest element in a tree and can therefore be counted in linear time (Sand, Holt, et al., 2013).
The triplet distance is the counterpart to the quartet distance. It is used to compute the distance between two rooted trees, where the triplet is the smallest informative substructure of a rooted tree (Sand, Holt, et al., 2013). The underlying algorithm equals the algorithm of the quartet distance. However, the triplet distance compensate the disadvantage of using unrooted trees of the quartet distance. The algorithm creates subsets of triplets out of the leaves and their corresponding labels. It counts the identical triplets in both trees. The distance is the number of different triplets in two trees (Sand, Holt, et al., 2013).
However, both distance measures also have disadvantages. The fastest algorithm of the triplet distance has a time complexity of $O(n \log n)$ for binary trees. The Robinson-Foulds distance can be computed in linear time ($O(n)$). This is known as the "optimal algorithmic complexity" (Sand, Holt, et al., 2013, p. 1190). The Robinson-Foulds distance can be computed faster than the triplet distance. This is one reason for the popularity of the Robinson-Foulds distance. Nevertheless, the triplet distance is improved from time to time and the current version provides also an efficient way to compute the distance between two trees. Another reason for the popularity of the Robinson-Foulds distance is their simplicity. However, the simplicity is also a disadvantage of the algorithm. The distance measurement is sensitive to outliers. This means that only a few changes in the set of leaves can have a significant impact on the output of the algorithm (Holt, Johansen, & Brodal, n.d.). This property might have a greater influence on the results as we can see in the chart in figure 5.2(a). For bigger trees, the probability that two trees differ increases. Having an algorithm which is sensible to outliers can cause unintentional results and can lead to a misinterpretation of evolutionary events. This is not the case for the triplet distance. This distance measure is "known to be more robust to small changes in the trees than other distance measures, including the Robinson-Foulds distance" (Sand, Holt, et al., 2013, p. 1192). This means that in contrast to the Robinson-Foulds distance, the triplet distance is more stable according to small changes. Thus, it is unlikely that the triplet distance causes unintentional results. Table 5.1 summarizes the advantages and disadvantages for both distance measures.

Robinson-Foulds distance	Triplet distance
runs faster	runs slower
sensitive to outliers	more robust to small changes

Table 5.1: Differences between the algorithms of the Robinson-Foulds and the triplet distance

The Robinson-Foulds distance can compute the distance between two trees faster than the triplet distance. Nevertheless, the triplet distance is more stable to small changes than the Robinson-Foulds distance. The stability of the algorithm to small changes within the trees, is more important than the faster runtime. The current algorithm for the triplet distance may not compute the distance in optimal time, but we can assume that it leads to better and more stable results. This advantage is more important for our purpose than the run time. Therefore, the triplet distance is the best distance measure to compute two rooted trees.
The implemented program stated above only uses the triplet distance to compare two trees. According to the distance and the threshold, the languages which cause an evolutionary event are found. The bar chart in figure 5.2(c) shows, that for this language sample only English causes an event. The program computes a network which visualizes the language and the reticulations.

5.2 Evaluation of the network

A network is a possibility to visualize a tree with reticulations. As explained in section 2.2, there are different types of networks within phylogenetics. There are two types of networks, unrooted and rooted. Unrooted networks are always data-display networks; rooted networks are mostly evolutionary networks.

Data-display networks only have the function to display the data and its structure. The program also computes an unrooted network to visualize the data.
The network displays the language sample of Germanic and Romance languages. It is computed using the underlying language tree of the language sample and represents the language causes an evolutionary event. The concept used for the computation is "mountain". The network shown in figure 5.3 displays the structure of the language sample and additionally the reticulation of English. English is connected to its parent node in the language tree with the blue line and to its corresponding sister node in the concept tree with the red line. We need to keep in mind that this abstract network is not able to represent the evolutionary history of the languages. Therefore, a rooted network is needed.

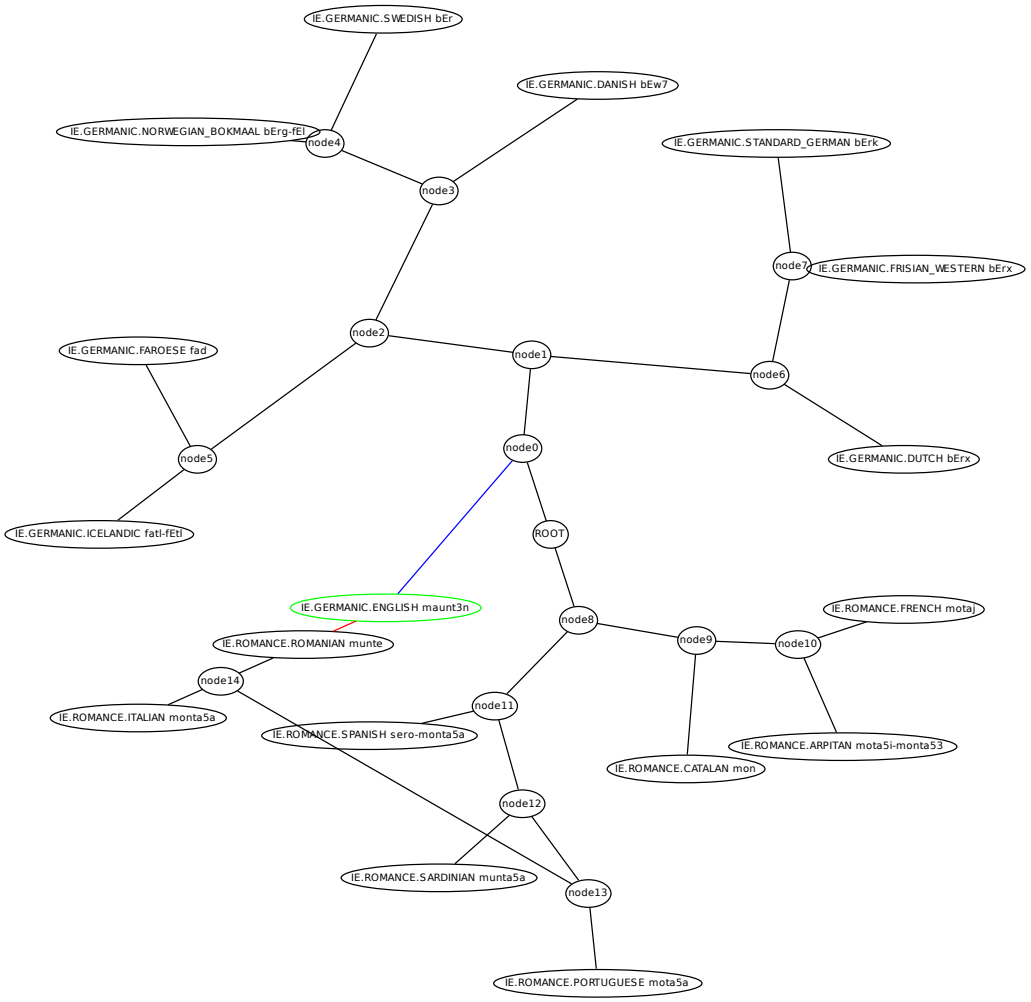


Figure 5.3: An unrooted network of Germanic and Romance languages for the concept "mountain"

Evolutionary networks are able to represent a language history. The root indicates the split of the two language families. The edges express the history of the languages within the tree. A split of one language into two languages is specified by an inner node. The reticulations mark the connection between two languages. They can be either unrelated or within the same language family. The reticulations within the network represent the evolutionary events which happened during language evolution.
The network is computed in a similar way than the unrooted. The underlying tree is the language tree of the language sample. The language which causes an evolutionary event is coloured green. The corresponding reticulations are also coloured. The connection between the language and its parent node in the language tree is coloured blue, whereas the reticulations to the concept tree are coloured red.

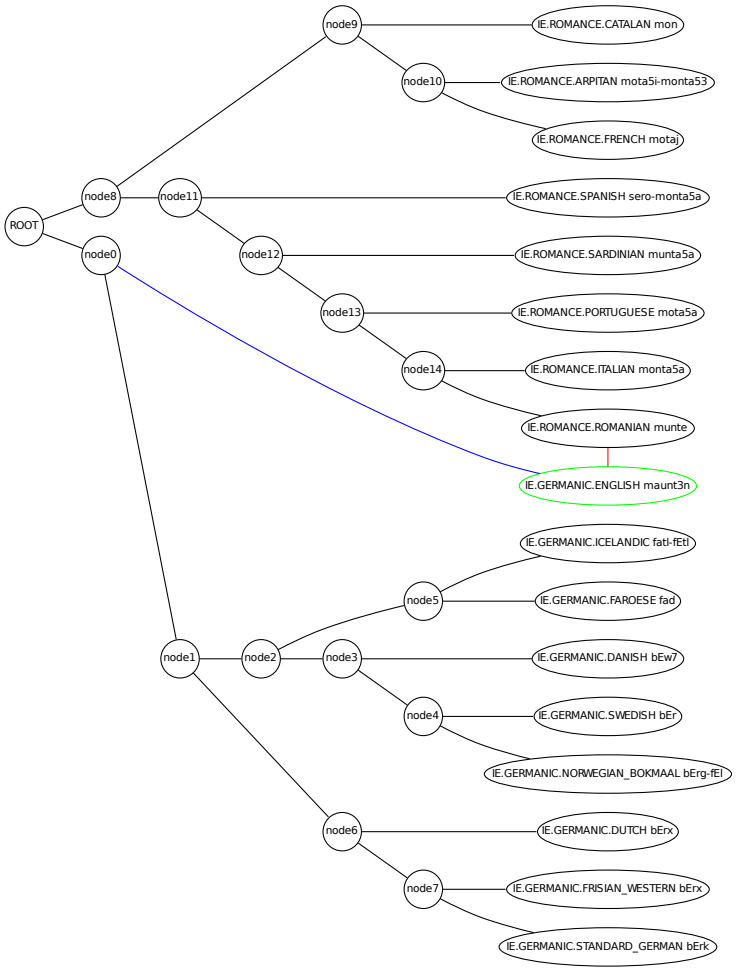


Figure 5.4: A rooted network of the Germanic and Romance languages for the concept "mountain"

The visualization of the results within the network are as expected. The underlying tree is responsible for the ordering of the languages. The language families are ordered in the corresponding way, namely all Romance and all Germanic languages together. The English language is grouped within the Germanic language family, but has a reticulation to the Romanian language. The Romanian language is the sister node to English in the concept tree. The red reticulation indicates the connection of English to the Romance language family. This connection is due to language contact and the following process of borrowing. English borrowed the word *mountain* from Old French *montaigne* as explained in section 3. This evolutionary event is indicated by the red line. The reason why English is connected to Romanian is due to the data. In the concept tree, English is a sister node to Romanian. The ASJP database does not contain Old French and the data therefore does not include this concept representations. The closest language where the concept representations for "mountain" is very much alike to

English is thus chosen. In this case, Romanian. Therefore, English is connected to Romanian.

The reticulation is the only one present within the network. This is a consequence of the computation of the threshold. The distance measurement and the threshold filter the languages which cause no significant difference between the trees. However, we need to keep in mind that the algorithm and the network can not distinguish between different kinds of events. The identification is left to interpretation, predictions and background knowledge. Therefore, the algorithm does not guarantee the detection of a specific sort of events, but sorts out the languages which cause no statistical difference. We minimize the number of evolutionary events to the significant events which can be interpreted more easily. This method improves the detection of the evolutionary events in contrast to the detection methods from phylogentics.

The algorithm is able to root trees on a specific outgroup. The outgroup is defined by the user who can specify the languages of the outgroup. The algorithm searches for the languages and their next common ancestor within the tree and roots it according to this node. It could be the case, that some other languages are included in the outgroup which were not defined. This is the case for the example in figure 5.5, which shows a language tree and a concept tree with the representation of "dog".

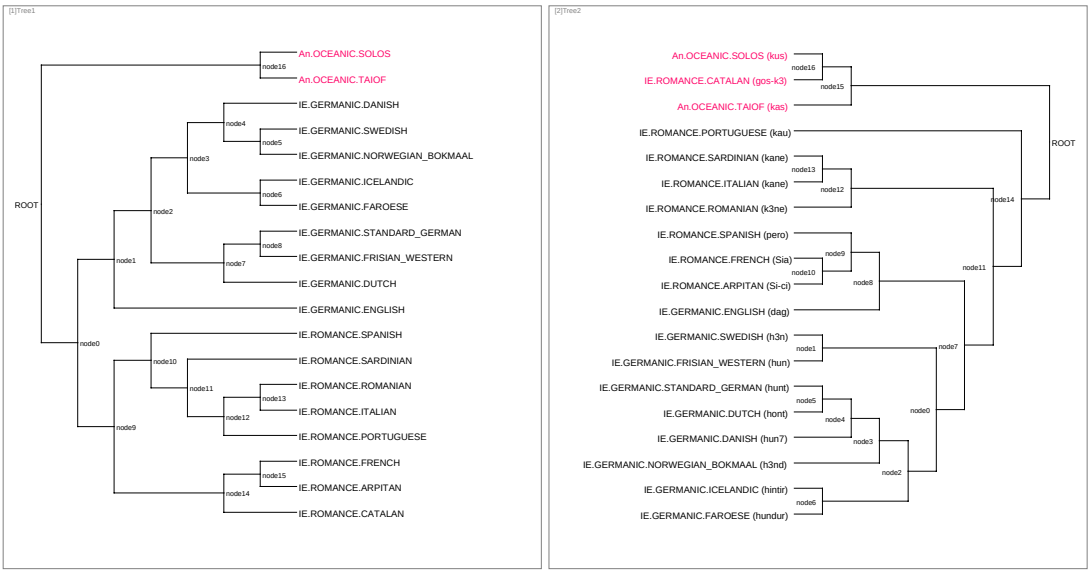


Figure 5.5: The language and concept tree for the concept "dog"

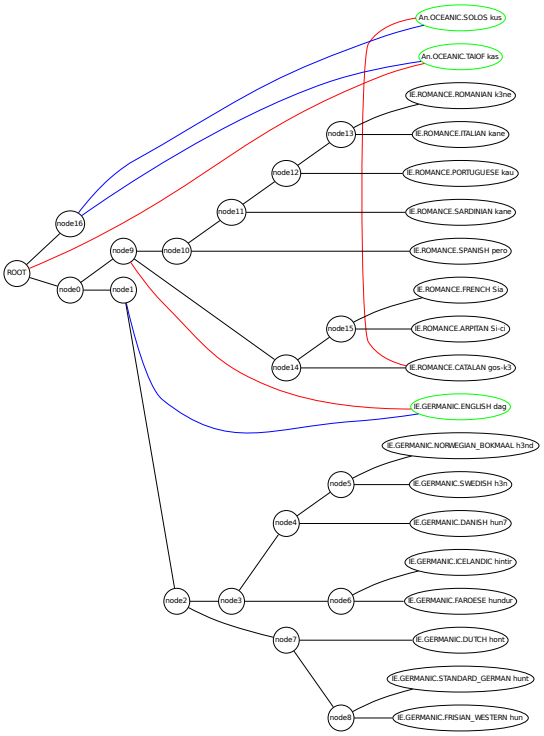
In figure 5.5, the defined outgroup are the two Austronesian languages Solos and Taiof. The language tree can be rooted on this outgroup correctly, whereas the concept tree is rooted on the languages plus Catalan. This is caused by the data. The language tree is created out of 40 concepts. A single concept

representation does not carry that much weight within the computation. The concept tree however is created with one single concept. This single representation is responsible for the grouping of the languages within the tree. Figure 5.5 shows, that the concept tree is rooted on the next common ancestor of the languages Solos and Taiof. Catalan is grouped within the two languages and is therefore also included in the outgroup. As it can be seen in figure 5.5, the representations of the concepts are similar. In this case, the Austronesian languages are not that different as one would them expect to be. The algorithm groups the languages in the correct way and the forming of the outgroup is therefore a consequence of the data used for the computation.

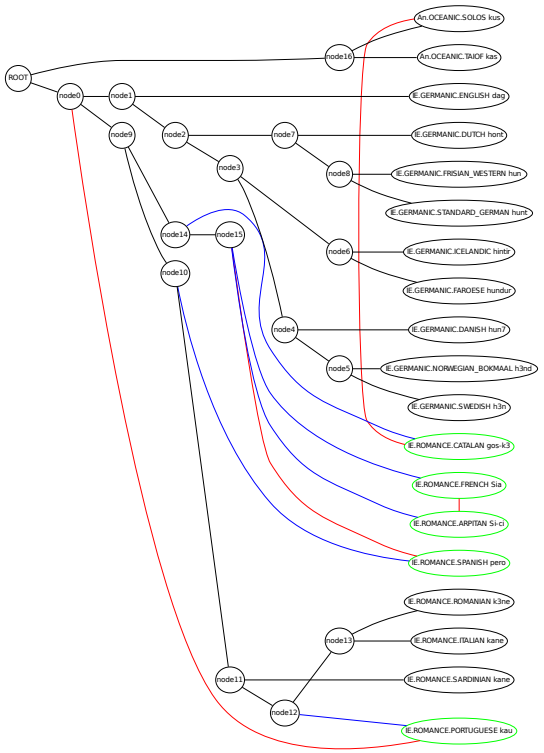
The question arises if this is the desired result or not and if not, what is the alternative? The alternative to detection of the common ancestor is to detach each language defined within the set of the outgroup and add them as children to the root. The advantage is that the tree can be rooted on the set of languages which was defined and other languages are left out. The disadvantage is that information can be lost. The ordering of the languages within the tree are due to the algorithms of constructing a distance matrix and a linguistic tree. Similar languages are ordered closer than distinct languages. With the detachment of nodes, the structure of the tree gets disarranged and a new way of ordering them is needed. Language trees are more resistant against this problem of rooting than concept trees. Therefore, the algorithm cannot guarantee the same outgroup in both trees. The user needs to be aware of that fact while using the program. Nevertheless, the rooting can have an influence on the network.

Figure 5.6(a) displays a rooted network for the language sample of Germanic, Romance and Astronesian languages and the concept of "dog" without a defined outgroup. In contrast figure 5.6(b) shows the network for the rooted trees in figure 5.5. Within the trees, the language tree is rooted on the two Austronesian languages, whereas the concept tree is rooted on the common ancestor of the two, including Catalan. The question arises if the rooted network is an improvement to the unrooted.

In the network in figure 5.6(a), English is marked as a language which causes an evolutionary event. However, according to the World Loanword Database (WOLD) Haspelmath and Tadmor (n.d.) the English word "dog" has no evidence for borrowing. It must therefore, cause another evolutionary event or is just due to the data. Within the network in figure 5.6(b), English is no longer causing an evolutionary event. However, within the Romance languages several events emerge which were not present within the other network. Those are phenomena which take place within a language family. These can be sound change, morphological change and borrowing. The algorithm cannot distinguish between



(a) A network without defined outgroup



(b) A network with defined outgroup

Figure 5.6: The rooted networks of the Germanic, Romance and Austronesian language sample for the concept "dog"

the different events and therefore, the identification is left to interpretation, predictions and background knowledge. The emergence of the events is caused by the rerooting of the concept tree.

Language	**Concept Representation**
Portuguese	*kau*
Spanish	*pero*
Catalan	*gos-k3*
French	*Sia*
Arpitan	*Si-ci*

Table 5.2: Languages causing an evolutionary event for the concept "dog"

Table 5.2 summarizes the languages and their corresponding representation of the concept "dog" which cause an evolutionary event according to the network. It can be seen that all languages have different representations. Catalan is connected to the Austronesian languages. The concept representation of Solos is *kus*. In the WOLD database, Haspelmath and Tadmor (n.d.) show that some Austronesian languages like Hawaiian had contact to Germanic and Romance languages. Hence, it may have been the case that Catalan and Solos were in contact. Other Romance language have similar representations of the concept, which is another evidence for the similarity of the concepts:

- Italian: *kane*
- Portuguese: *kau*
- Romanian: *k3ne*
- Sardinian: *kane*

The WOLD database contains two Austronesian languages from the same language family than Solos and Taiof, namely Hawaiian and Takia. The word "dog" in Hawaiian has no evidence for borrowing, but the "dog" in Takia is perhaps borrowed. Haspelmath and Tadmor (n.d.) do not specify the source language. Nevertheless, this is no evidence for a contact between Solos and Catalan. It can be the case, that the connection is due to convergent evolution.
The rooting of the trees is no great improvement of the network. It would only be an improvement if both trees can be rooted on the same outgroup. This can only be guaranteed if the user is aware of the input data and of setting an outgroup.

Another kind of rooting is to specify only one language as an outgroup. In phylogenetics, it is mostly the case that an outgroup contains only one species. This can be adapted into linguistics, as it can be seen in figure 5.7.

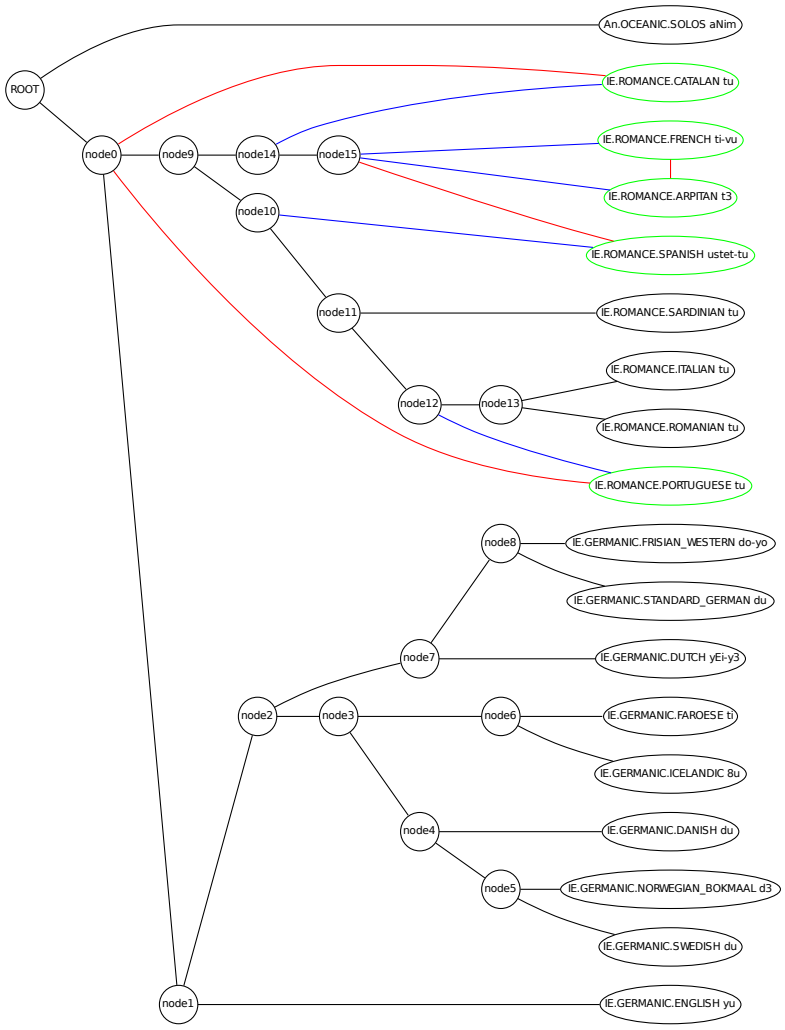


Figure 5.7: A rooted network on one language

The network is computed out of a language tree and a concept tree with the representation of "dog". If only one language serves as an outgroup, the trees are rooted according to the leaf node of the language. In this case, the Austronesian language Solos was specified as outgroup. The algorithm roots the trees on this leaf node. With this technique, both trees can be rooted the same way. This is due to the fact, that the algorithm does not search for the next common ancestor but only for the leaf node. Other languages within the outgroup can be prevented. The structure of the trees changes, but the possibility of losing information is smaller than within a bigger outgroup. In contrast to the network shown in figure 5.6(b), Solos causes no evolutionary event. The other reticulations present in figure 5.7 are evolutionary events within one language family. Those reticulation equal the reticulation within the network in figure 5.6(b). Therefore, the same evolutionary events are depicted in both networks. This gives us evidence that

the rerooting is an improvement of the network. Nevertheless, the user need to be aware of the data and the outgroup. It need to be clear, if the data enables a rerooting or not.
All in all it can be said that the networks represented in this section are an improvement of the networks constructed with phylogenetic methods. It comes closer to the approach of an evolutionary network from Morrison (2011). The algorithm leads to correct results and minimizes the number of reticulations. We need to keep in mind, that the data plays a crucial role to construct the distance matrices, the trees and the network. Without sufficient data, the algorithms might lead to unexpected results. However, it can detect evolutionary events using the ASJP database and the corresponding representations of the concepts. The network is not able to distinguish between different evolutionary events. The identification is due to interpretations, predictions and background knowledge. Nonetheless, with linguistic background knowledge expected and unexpected results can be visualized and this is a great improvement in the adaption of phylogenetic methods to linguistics.

6 Conclusion

The paper emphasises the existing connection between pyhlogentics and linguistcs and especially the use and adaption of phylogenentic methods within historical linguistics. Several methods can be used without modification. Nevertheless, the modification and integration of other methods improves the results on linguistic data.

The idea introduced in this paper is the detection of mismatches between two linguistic trees. Linguistic trees can either be language trees or concept trees. The data used for the reconstruction of these trees is provided by the ASJP database (Wichmann et al., 2012). A language tree represents the history of the languages. All concept representations of the languages are used for the reconstruction. A concept tree, on the other hand, indicates the history of one single concept. Therefore, only one concept representation of the languages is used for the creation of such a tree. A concept tree can be mapped to a language tree to compute their mismatch. The mismatch of the trees can be measured in several ways. Firstly, algorithms from the phylogenetic program Dendroscope can be used. These algorithms construct a phylogenetic network which is able to visualize the mismatch. Another approach is the detection of a mismatch using distances. A single distance measurement cannot locate the source of the difference between the trees. Therefore, one language at a time is removed from the language and concept trees. For these trees, the distance is measured. If the distance gets smaller, the language had a great impact on the original tree structure. To distinguish between languages which cause a significant distance and those which are not, a threshold is computed. Only those languages which are under this threshold cause a significant distance and therefore a mismatch. The mismatch is indicated by reticulations which can be interpreted as evolutionary events. Evolutionary events can occur within a language family and between two language families. To visualize these events, a network is drawn. The network is able to depict reticulations and therefore evolutionary events. All evolutionary events within a language family are listed in H.2 for the Germanic and Romance language sample. Additionally for the same sample, the evolutionary between language families are listed in H.1.

The algorithm is not able to distinguish between different events. The identification is left to interpretation, prediction or background knowledge. Therefore, further work is needed to classify these events.

Nevertheless, the comparison of trees using distances is an adequate method to detect a mismatch. This mismatch gives us an insight to the evolution of language history and the corresponding events occurring during this period.

References

Association, I. P. (1999). *Handbook of the international phonetic association: A guide to the use of the international phonetic alphabet*. Cambridge University Press.

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.

Baayen, R. H. (2008). *Analyzing linguistic data*.

Baldauf, S. L. (2003). Phylogeny for the faint of heart: a tutorial. *TRENDS in Genetics*, *19*(6), 345–351.

Brodal, G. S., Fagerberg, R., Mailund, T., Pedersen, C. N., & Sand, A. (2013). Efficient algorithms for computing the triplet and quartet distance between trees of arbitrary degree. In *Soda* (pp. 1814–1832).

Brodal, G. S., Fagerberg, R., & Pedersen, C. N. (2004). Computing the quartet distance between evolutionary trees in time o (n log n). *Algorithmica*, *38*(2), 377–395.

Brown, C. H., Holman, E. W., Wichmann, S., & Velupillai, V. (2008). Automated classification of the world's languages: a description of the method and preliminary results. *STUF-Language Typology and Universals*, *61*(4), 285–308.

Bryant, D., Tsang, J., Kearney, P., & Li, M. (2000). Computing the quartet distance between evolutionary trees. In *Proceedings of the eleventh annual acm-siam symposium on discrete algorithms* (pp. 285–286).

Bußmann, H. (Ed.). (2008). *Lexikon der sprachwissenschaft: mit ...14 tabellen ...* Stuttgart: Kröner.

Campbell, L. (2013). *Historical linguistics: an introduction* (3. ed. ed.). Cambridge, Mass.: MIT Press.

Critchlow, D. E., Pearl, D. K., & Qian, C. (1996). The triples distance for rooted bifurcating phylogenetic trees. *Systematic Biology*, *45*(3), 323–334.

Crowley, T., & Bowern, C. (2010). *An introduction to historical linguistics* (4. ed. ed.). Oxford: Oxford University Press. Available from `http://digitool.hbz-nrw.de:1801/webclient/DeliveryManager?pid=3741953`

Darwin, C. (1871). *The descent of man.* D. Appleton and Company.

Delz, M., Layer, B., Schulz, S., & Wahle, J. (2012, March). Overgeneralisation of verbs - the change of the german verb system. In *Proceedings of the 9th international conference on the evolution of language* (p. 96-103). Kyoto, Japan.

Desper, R., & Gascuel, O. (2002). Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *Journal of computational biology*, *9*(5), 687–705.

Doolittle, W. F. (1999). Phylogenetic classification and the universal tree. *Science*, *284*(5423), 2124–2128.

Dryer, M. S., & Haspelmath, M. (Eds.). (2013). *Wals online.* Leipzig: Max Planck Institute for Evolutionary Anthropology. Available from `http://wals.info/`

Eldredge, N. (2005). *Darwin: discovering the tree of life.* New York [u.a.]: Norton.

Estabrook, G. F., McMorris, F., & Meacham, C. A. (1985). Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. *Systematic Biology*, *34*(2), 193–200.

Felsenstein, J. (2004). *Inferring phylogenies.* Sunderland, Mass.: Sinauer. Available from `http://www.ulb.tu-darmstadt.de/tocs/103801863.pdf`

Forster, P. (2006). *Phylogenetic methods and the prehistory of languages.* McDonald Institute for Archaeological Research.

Gascuel, O. (1997a). Bionj: an improved version of the nj algorithm based on a simple model of sequence data. *Molecular biology and evolution*, *14*(7), 685–695.

Gascuel, O. (1997b). Concerning the nj algorithm and its unweighted version, unj. *Mathematical hierarchies and biology*, *37*, 149–171.

Haeckel, E. H. P. A. (1874). *Anthropogenie oder entwicklungsgeschichte des menschen.* Leipzig: Verlag von Wilhelm Engelmann.

Hammarström, H. (2010). A full-scale test of the language farming dispersal hypothesis. *Diachronica*, *27*(2), 197–213.

Harper, D. (n.d.). *Online etymology dictionary.* Available from `http://www.etymonline.com/index.php`

Haspelmath, M., & Tadmor, U. (n.d.). *World loanword database.* Available from `http://wold.livingsources.org/`

Haspelmath, M., & Tadmor, U. (2009). *Loanwords in the world's languages: a comparative handbook.* Walter de Gruyter.

Haugen, E. (1950). The analysis of linguistic borrowing. *Language*, *26*(2), 210–231.

Heggarty, P. (2006). Interdisciplinary indiscipline? can phylogenetic methods meaningfully be applied to language data—and to dating language. *Phylogenetic methods and the prehistory of languages. Cambridge*, 183–94.

Holden, C. J., & Gray, R. D. (2006). Rapid radiation, borrowing and dialect continua in the bantu languages. *Phylogenetic methods and the prehistory of languages*, 19–31.

Holt, M. K., Johansen, J., & Brodal, G. S. (n.d.). On the scalability of computing triplet and quartet distances. In *Workshop on algorithm engineering and experiments* (pp. 9–19).

Huson, D. H., Richter, D. C., Rausch, C., Dezulian, T., Franz, M., & Rupp, R. (2007). Dendroscope: An interactive viewer for large phylogenetic trees. *BMC bioinformatics*, *8*(1), 460.

Huson, D. H., Richter, D. C., Rausch, C., & Rupp, R. (2010). User manual for dendroscope v2. 7.4.

Huson, D. H., & Rupp, R. (2008). Summarizing multiple gene trees using cluster networks. In *Algorithms in bioinformatics* (pp. 296–305). Springer.

Huson, D. H., Rupp, R., & Scornavacca, C. (2010). *Phylogenetic networks: concepts, algorithms and applications.* Cambridge University Press.

Huson, D. H., & Scornavacca, C. (2012). Dendroscope 3: an interactive tool for rooted phylogenetic trees and networks. *Systematic biology*, *61*(6), 1061–1067.

Jackendoff, R. (1999). Possible stages in the evolution of the language capacity. *Trends in cognitive sciences*, *3*(7), 272–279.

Jäger, G. (2013). *Evaluating distance-based pyhlogenetic algorithms for automated language classification.*

Jäger, G. (2013). Phylogenetic inference from word lists using weighted alignment with empirically determined weights. *Language Dynamics and Change.*

Joseph, B. D., & Janda, R. D. (2003). *The handbook of historical linguistics.* Wiley Online Library.

Kessler, B., & Lehtonen, A. (2006). Multilateral comparison and significance testing of the indo-uralic question. *Phylogenetic Methods and the Prehistory of Languages. Cambridge, England: McDonald Institute for Archaeological Research*, 33–42.

Lecointre, G. (2006). *The tree of life: a phylogenetic classification* (H. Le Guyader, Ed.). Cambridge, MA: Belknap Press of Harvard Univ. Pr.

Lewis, M. P. (2009). Ethnologue: Languages of the world sixteenth edition. *Dallas, Tex.: SIL International. Online version: http://www. ethnologue. com.*

List, J.-M. (2013). Improving phylogeny-based network approaches to investigate the history of the chinese dialects. In *Lfk society young scholars symposium.*

Mailund, T., & Pedersen, C. N. (2004). Qdist—quartet distance between evolutionary trees. *Bioinformatics*, *20*(10), 1636–1637.

McMahon, A., & McMahon, R. (2005). *Language classification by numbers.* Oxford University Press.

McMahon, A. M. S. (1995). *Understanding language change* (Repr. ed.). Cambridge [u.a.]: Cambridge Univ. Press. Available from 04

Morrison, D. A. (2011). *Introduction to phylogenetic networks.* Uppsala, Sweden: RJR Productions.

Nichols, R. (2001). Gene trees and species trees are not the same. *Trends in Ecology & Evolution*, *16*(7), 358–364.

Nowak, M. A., & Krakauer, D. C. (1999). The evolution of language. *Proceedings of the National Academy of Sciences*, *96*(14), 8028–8033.

Page, R. D., & Charleston, M. A. (1997). From gene to organismal phylogeny: reconciled trees and the gene tree/species tree problem. *Molecular phylogenetics and evolution*, *7*(2), 231–240.

Penny, D. (2011). Darwin's theory of descent with modification, versus the biblical tree of life. *PLoS biology*, *9*(7), e1001096.

Robinson, D., & Foulds, L. (1979). Comparison of weighted labelled trees. In *Combinatorial mathematics vi* (pp. 119–126). Springer.

Robinson, D., & Foulds, L. R. (1981). Comparison of phylogenetic trees. *Mathematical Biosciences*, *53*(1), 131–147.

Robinson, D. F. (1971). Comparison of labeled trees with valency three. *Journal of Combinatorial Theory, Series B*, *11*(2), 105–119.

Sand, A., Brodal, G. S., Fagerberg, R., Pedersen, C. N., & Mailund, T. (2013). A practical o (n log2 n) time algorithm for computing the triplet distance on binary trees. *BMC bioinformatics*, *14*(Suppl 2), S18.

Sand, A., Holt, M. K., Johansen, J., Fagerberg, R., Brodal, G. S., Mailund, T., et al. (2014). tqdist: A library for computing the quartet and triplet distances between binary or general trees. *BMC Bioinformatics*.

Sand, A., Holt, M. K., Johansen, J., Fagerberg, R., Brodal, G. S., Pedersen, C. N., et al. (2013). Algorithms for computing the triplet and quartet distances for binary and general trees. *Biology*, *2*(4), 1189–1209.

Schleicher, A. (1873). *Die darwinsche theorie und die sprachwissenschaft* (2. ed. ed.). Weimar: Hermann Böhlau.

Schülke, T. (2005). Der mittelniederdeutsch-skandinavische sprachkontakt zur hansezeit (1300-1550).

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

Warnow, T., Evans, S. N., Ringe, D., & Nakhleh, L. (2006). A stochastic model of language evolution that incorporates homoplasy and borrowing. *Phylogenetic methods and the prehistory of languages*, 75–90.

Wichmann, S., Müller, A., Velupillai, V., Wett, A., Brown, C. H., Molochieva, Z., et al. (2012). *The asjp database.* Available from `http://wwwstaff.eva.mpg.de/ wichmann/ASJPHomePage.htm`

Wiley, E. O., & Lieberman, B. S. (2011). *Phylogenetics: theory and practice of phylogenetic systematics.* John Wiley & Sons.

A The Swadesh 100-word list

Swadesh's 100-word list

1. I
2. thou
3. we
4. this
5. that
6. who?
7. what?
8. not
9. all
10. many
11. one
12. two
13. big
14. long
15. small
16. woman
17. man
18. person
19. fish
20. bird
21. dog
22. louse
23. tree
24. seed
25. leaf
26. root
27. bark
28. skin
29. flesh
30. blood
31. bone
32. grease
33. egg
34. horn
35. tail
36. feather
37. hair
38. head
39. ear
40. eye
41. nose
42. mouth
43. tooth
44. tongue
45. fingernail
46. foot
47. knee
48. hand
49. belly
50. neck
51. breasts
52. heart
53. liver
54. drink
55. eat
56. bite
57. see
58. hear
59. know
60. sleep
61. die
62. kill
63. swim
64. fly
65. walk
66. come
67. lie
68. sit
69. stand
70. give
71. say
72. sun
73. moon
74. star
75. water
76. rain
77. stone
78. sand
79. earth
80. cloud
81. smoke
82. fire
83. ash
84. burn
85. path
86. mountain
87. red
88. green
89. yellow
90. white
91. black
92. night
93. hot
94. cold
95. full
96. new
97. good
98. round
99. dry
100. name

B ASJP Orthography

VOWELS (symbols, modifiers, and conventions):
Symbols:
i = high front vowel, rounded and unrounded [IPA: i, I, y, ɣ]
e = mid front vowel, rounded and unrounded [IPA: e, ø]
E = low front vowel, rounded and unrounded [IPA: a, æ, ɛ, œ, œ]
3 = high and mid central vowel, rounded and unrounded [IPA: ɨ, ɘ, ə, ɜ, ʉ, ɵ, ɞ]
a = low central vowel, unrounded [IPA: ɐ]
u = high back vowel, rounded and unrounded [IPA: ɯ, u]
o = mid and low back vowel, rounded and unrounded [IPA: ɣ, ʌ, ɑ, o, ɔ, ɒ]

Modifier:
An asterisk (*) following any one of the above seven vowel symbols indicates vowel nasalization, for example, ta*k. ASJP judges nasalized vowels as being similar to their non-nasalized counterparts.

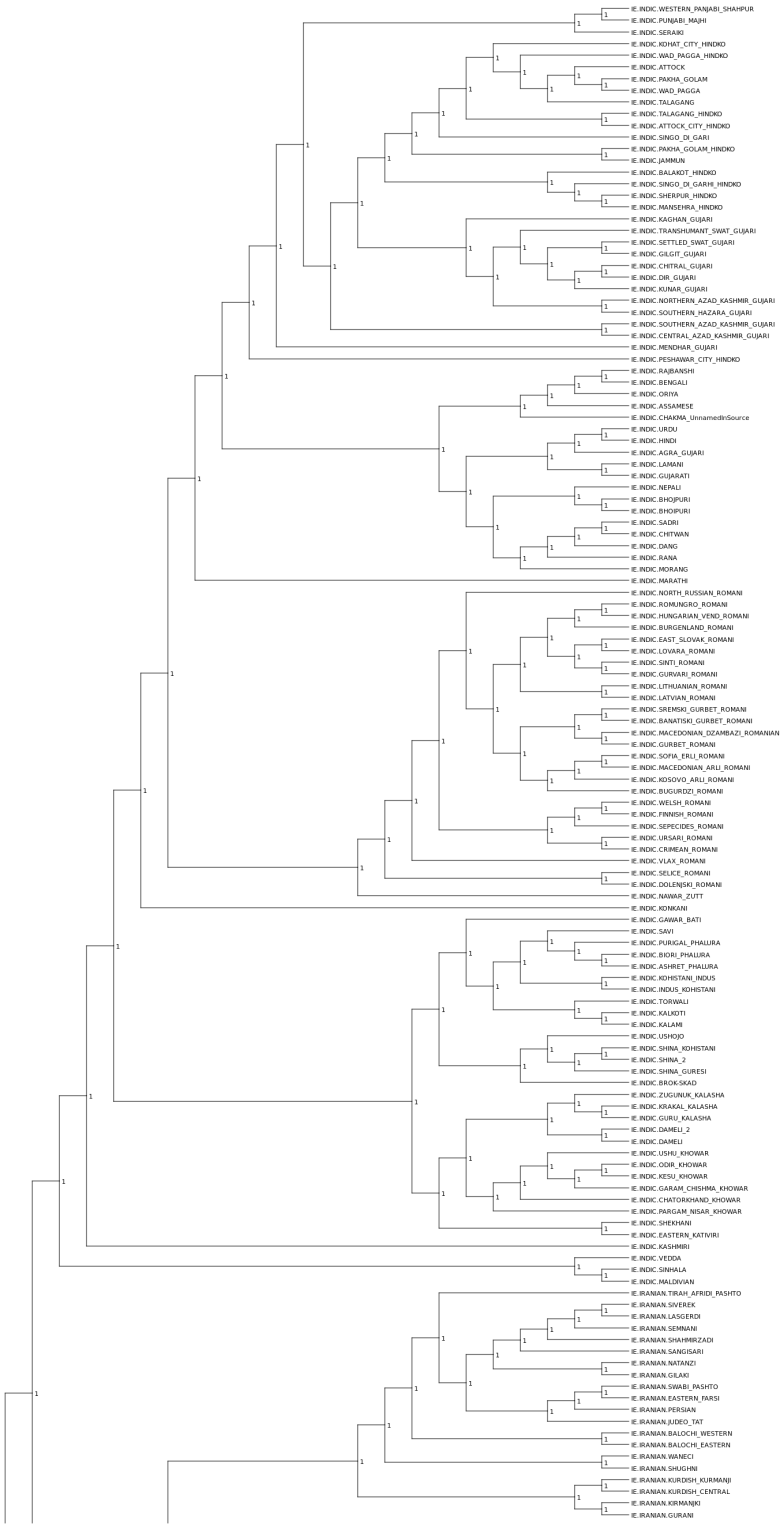
CONSONANTS (symbols, modifiers, and conventions):
Symbols:
p = voiceless bilabial stop and fricative [IPA: p, ɸ]
b = voiced bilabial stop and fricative [IPA: b, β]
m = bilabial nasal [IPA: m]
f = voiceless labiodental fricative [IPA: f]
v = voiced labiodental fricative [IPA: v]
8 = voiceless and voiced dental fricative [IPA: θ, ð]
4 = dental nasal [IPA: n̪]
t = voiceless alveolar stop [IPA: t]
d = voiced alveolar stop [IPA: d]
s = voiceless alveolar fricative [IPA: s]
z = voiced alveolar fricative [IPA: z]
c = voiceless and voiced alveolar affricate [IPA: ts, dz]
n = voiceless and voiced alveolar nasal [IPA: n]
S = voiceless postalveolar fricative [IPA: ʃ]
Z = voiced postalveolar fricative [IPA: ʒ]
C = voiceless palato-alveolar affricate [IPA: tʃ]
j = voiced palato-alveolar affricate [IPA: dʒ]
T = voiceless and voiced palatal stop [IPA: c, ɟ]
5 = palatal nasal [IPA: ɲ]
k = voiceless velar stop [IPA: k]
g = voiced velar stop [IPA: g]
x = voiceless and voiced velar fricative [IPA: x, ɣ]
N = velar nasal [IPA: ŋ]
q = voiceless uvular stop [IPA: q]
G = voiced uvular stop [IPA: G]
X = voiceless and voiced uvular fricative, voiceless and voiced pharyngeal fricative [IPA: χ, ʁ, ħ, ʕ]
7 = voiceless glottal stop [IPA: ʔ]
h = voiceless and voiced glottal fricative [IPA: h, ɦ]
l = voiced alveolar lateral approximate [IPA: l]
L = all other laterals [IPA: L, ɭ, ʎ]
w = voiced bilabial-velar approximant [IPA: w]
y = palatal approximant [IPA: j]
r = voiced apico-alveolar trill and all varieties of “r-sounds” [IPA: r, R, etc.]
! = all varieties of “click-sounds” [IPA: !, ǀ, ǁ, ǂ]

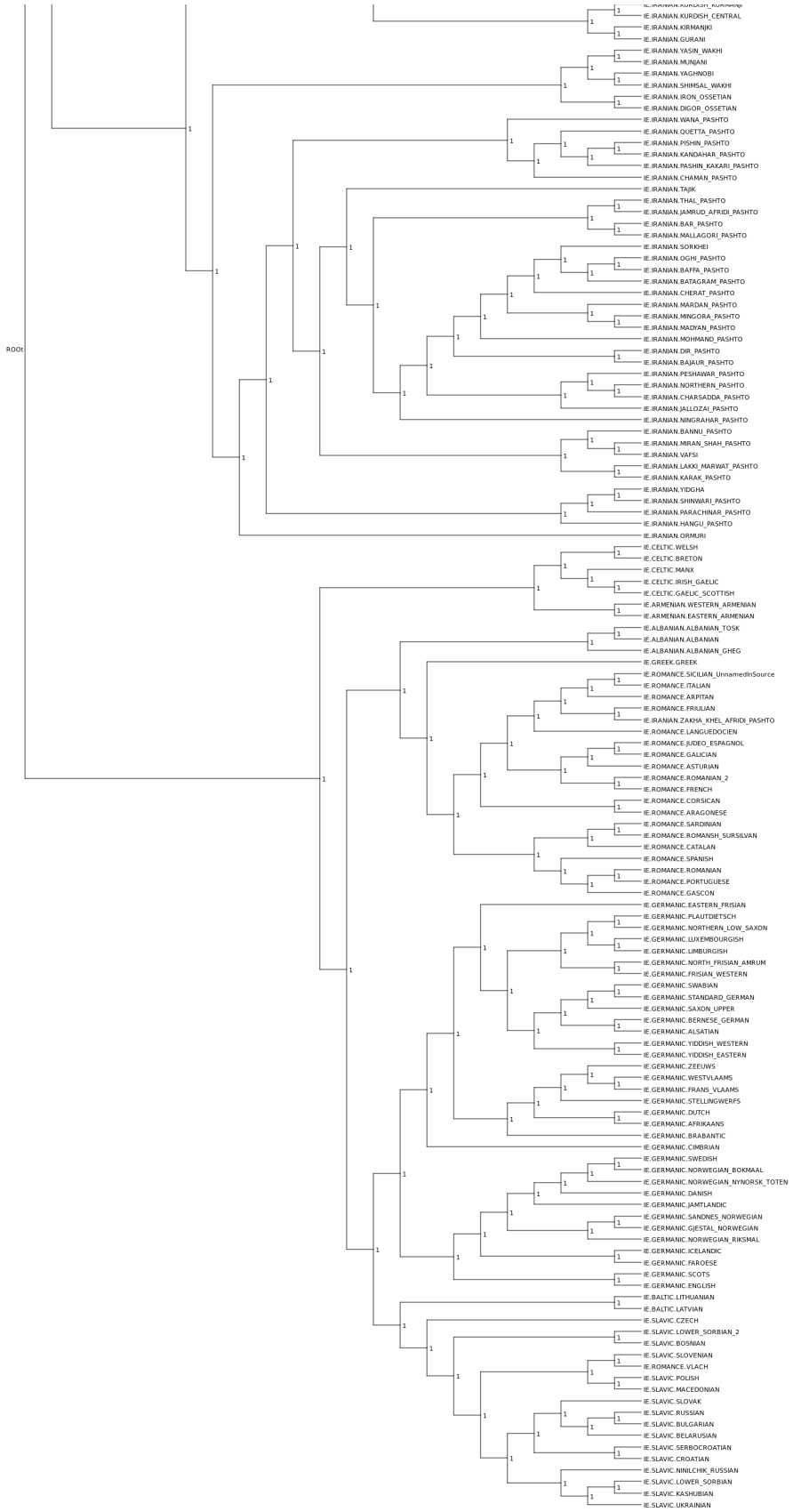
Modifiers:
The symbol ~ is a modifier that follows two juxtaposed consonants. ASJP regards such consonants as being in the same single position in a syllable. For example, kw~at is an ASJP transcription of a syllable originally transcribed by kwat. ASJP judges syllables such

C The ASJP 40-word list

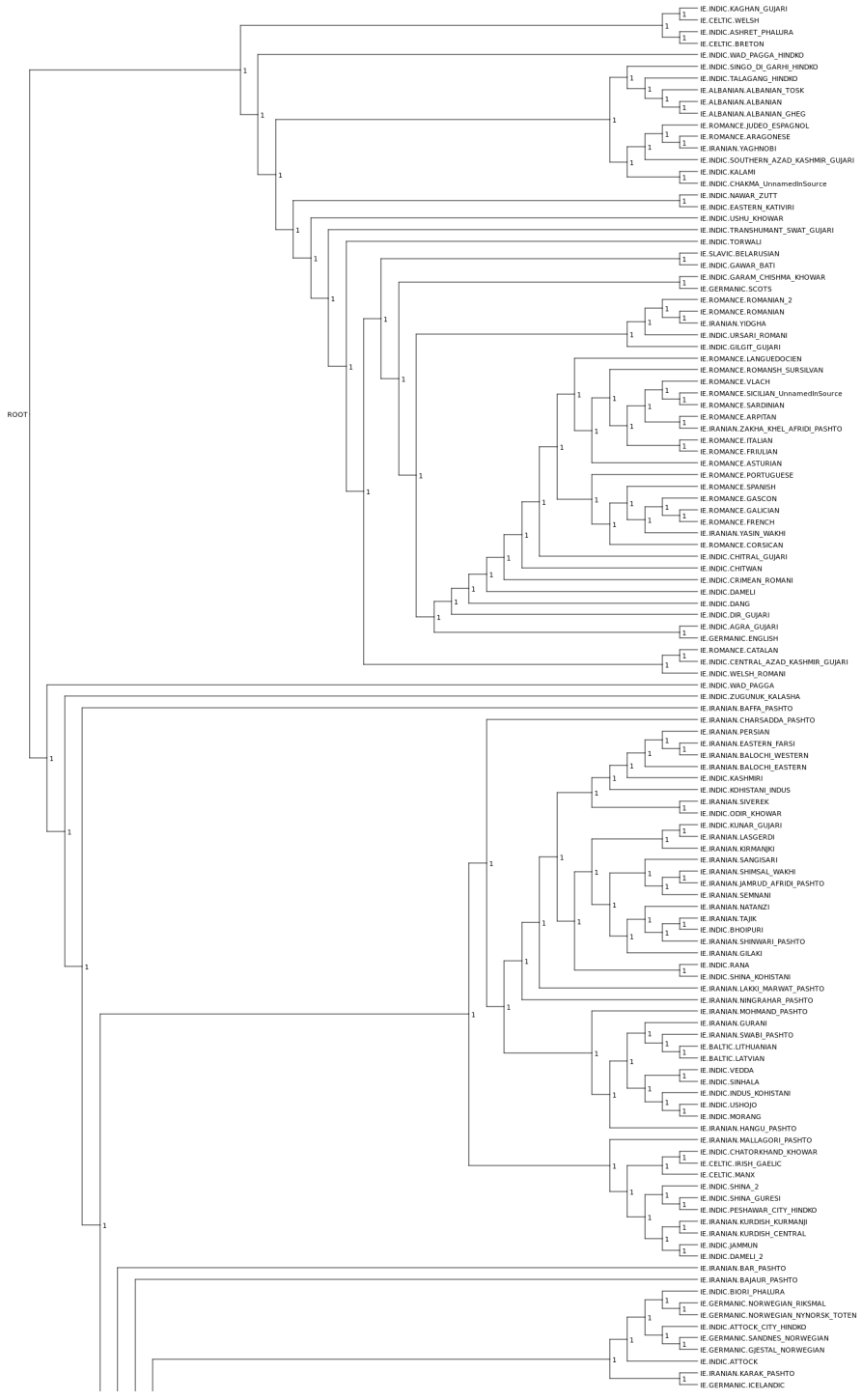
1. i
2. thou
3. we
4. one
5. two
6. person
7. fish
8. dog
9. louse
10. tree
11. leaf
12. skin
13. blood
14. bone
15. horn
16. ear
17. eye
18. nose
19. tooth
20. tongue
21. knee
22. hand
23. breast
24. liver
25. drink
26. see
27. hear
28. die
29. come
30. sun
31. star
32. water
33. stone
34. fire
35. path
36. mountain
37. night
38. full
39. new
40. name

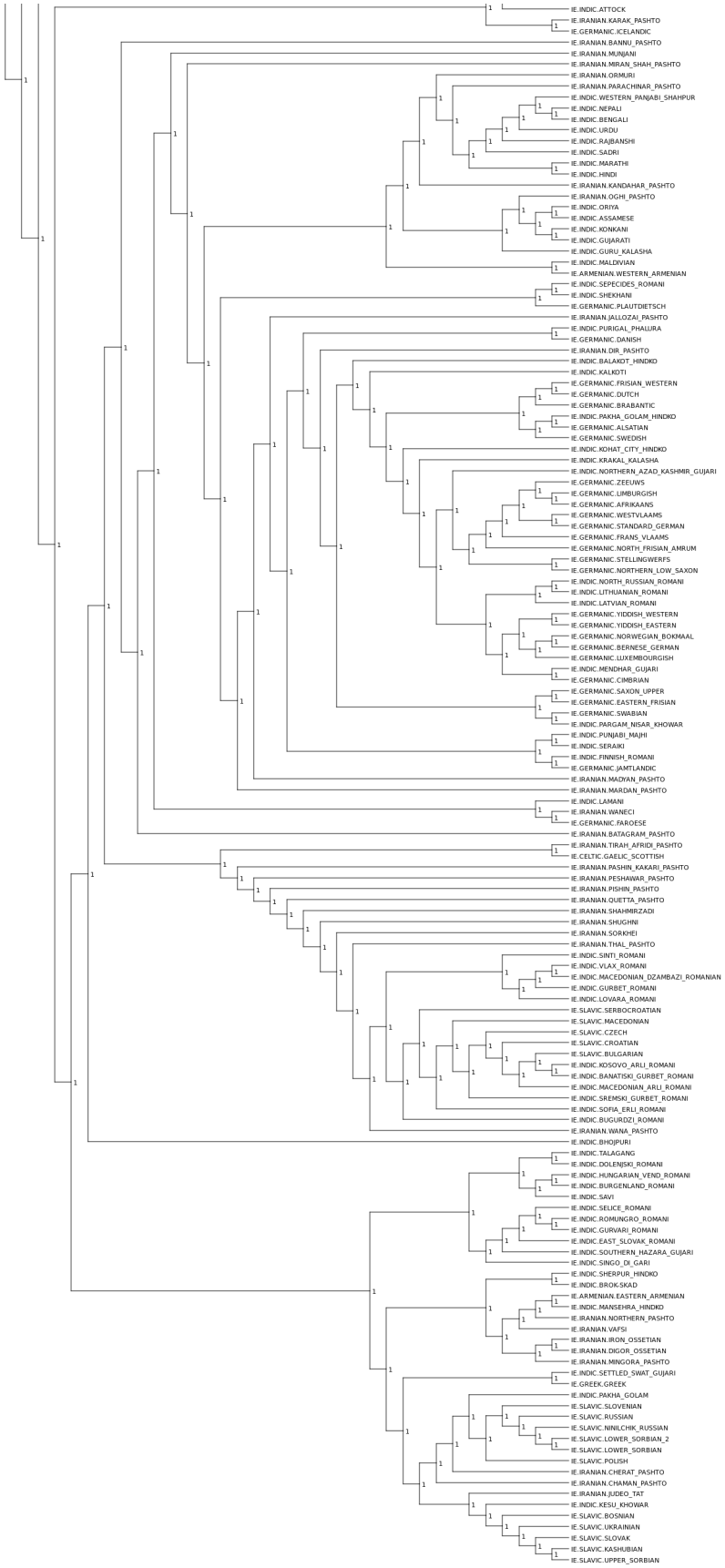
D Language Tree of the Indo-European languages




ROOT
IE.IRANIAN.KURDISH_CENTRAL
IE.IRANIAN.KIRMANJKI
IE.IRANIAN.GURANI
IE.IRANIAN.YASIN_WAKHI
IE.IRANIAN.MUNJANI
IE.IRANIAN.YAGHNOBI
IE.IRANIAN.SHIMSAL_WAKHI
IE.IRANIAN.IRON_OSSETIAN
IE.IRANIAN.DIGOR_OSSETIAN
IE.IRANIAN.WANA_PASHTO
IE.IRANIAN.QUETTA_PASHTO
IE.IRANIAN.PISHIN_PASHTO
IE.IRANIAN.KANDAHAR_PASHTO
IE.IRANIAN.PASHIN_KAKARI_PASHTO
IE.IRANIAN.CHAMAN_PASHTO
IE.IRANIAN.TAJIK
IE.IRANIAN.THAL_PASHTO
IE.IRANIAN.JAMRUD_AFRIDI_PASHTO
IE.IRANIAN.BAR_PASHTO
IE.IRANIAN.MALLAGORI_PASHTO
IE.IRANIAN.SORKHEI
IE.IRANIAN.OGHI_PASHTO
IE.IRANIAN.BAFFA_PASHTO
IE.IRANIAN.BATAGRAM_PASHTO
IE.IRANIAN.CHERAT_PASHTO
IE.IRANIAN.MARDAN_PASHTO
IE.IRANIAN.MINGORA_PASHTO
IE.IRANIAN.MADYAN_PASHTO
IE.IRANIAN.MOHMAND_PASHTO
IE.IRANIAN.DIR_PASHTO
IE.IRANIAN.BAJAUR_PASHTO
IE.IRANIAN.PESHAWAR_PASHTO
IE.IRANIAN.NORTHERN_PASHTO
IE.IRANIAN.CHARSADDA_PASHTO
IE.IRANIAN.JALLOZAI_PASHTO
IE.IRANIAN.NINGRAHAR_PASHTO
IE.IRANIAN.BANNU_PASHTO
IE.IRANIAN.MIRAN_SHAH_PASHTO
IE.IRANIAN.VAFSI
IE.IRANIAN.LAKKI_MARWAT_PASHTO
IE.IRANIAN.KARAK_PASHTO
IE.IRANIAN.YIDGHA
IE.IRANIAN.SHINWARI_PASHTO
IE.IRANIAN.PARACHINAR_PASHTO
IE.IRANIAN.HANGU_PASHTO
IE.IRANIAN.ORMURI
IE.CELTIC.WELSH
IE.CELTIC.BRETON
IE.CELTIC.MANX
IE.CELTIC.IRISH_GAELIC
IE.CELTIC.GAELIC_SCOTTISH
IE.ARMENIAN.WESTERN_ARMENIAN
IE.ARMENIAN.EASTERN_ARMENIAN
IE.ALBANIAN.ALBANIAN_TOSK
IE.ALBANIAN.ALBANIAN
IE.ALBANIAN.ALBANIAN_GHEG
IE.GREEK.GREEK
IE.ROMANCE.SICILIAN_UnnamedInSource
IE.ROMANCE.ITALIAN
IE.ROMANCE.ARPITAN
IE.ROMANCE.FRIULIAN
IE.IRANIAN.ZAKHA_KHEL_AFRIDI_PASHTO
IE.ROMANCE.LANGUEDOCIEN
IE.ROMANCE.JUDEO_ESPAGNOL
IE.ROMANCE.GALICIAN
IE.ROMANCE.ASTURIAN
IE.ROMANCE.ROMANIAN_2
IE.ROMANCE.FRENCH
IE.ROMANCE.CORSICAN
IE.ROMANCE.ARAGONESE
IE.ROMANCE.SARDINIAN
IE.ROMANCE.ROMANSH_SURSILVAN
IE.ROMANCE.CATALAN
IE.ROMANCE.SPANISH
IE.ROMANCE.ROMANIAN
IE.ROMANCE.PORTUGUESE
IE.ROMANCE.GASCON
IE.GERMANIC.EASTERN_FRISIAN
IE.GERMANIC.PLAUTDIETSCH
IE.GERMANIC.NORTHERN_LOW_SAXON
IE.GERMANIC.LUXEMBOURGISH
IE.GERMANIC.LIMBURGISH
IE.GERMANIC.NORTH_FRISIAN_AMRUM
IE.GERMANIC.FRISIAN_WESTERN
IE.GERMANIC.SWABIAN
IE.GERMANIC.STANDARD_GERMAN
IE.GERMANIC.SAXON_UPPER
IE.GERMANIC.BERNESE_GERMAN
IE.GERMANIC.ALSATIAN
IE.GERMANIC.YIDDISH_WESTERN
IE.GERMANIC.YIDDISH_EASTERN
IE.GERMANIC.ZEEUWS
IE.GERMANIC.WESTVLAAMS
IE.GERMANIC.FRANS_VLAAMS
IE.GERMANIC.STELLINGWERFS
IE.GERMANIC.DUTCH
IE.GERMANIC.AFRIKAANS
IE.GERMANIC.BRABANTIC
IE.GERMANIC.CIMBRIAN
IE.GERMANIC.SWEDISH
IE.GERMANIC.NORWEGIAN_BOKMAAL
IE.GERMANIC.NORWEGIAN_NYNORSK_TOTEN
IE.GERMANIC.DANISH
IE.GERMANIC.JAMTLANDIC
IE.GERMANIC.SANDNES_NORWEGIAN
IE.GERMANIC.GJESTAL_NORWEGIAN
IE.GERMANIC.NORWEGIAN_RIKSMAL
IE.GERMANIC.ICELANDIC
IE.GERMANIC.FAROESE
IE.GERMANIC.SCOTS
IE.GERMANIC.ENGLISH
IE.BALTIC.LITHUANIAN
IE.BALTIC.LATVIAN
IE.SLAVIC.CZECH
IE.SLAVIC.LOWER_SORBIAN_2
IE.SLAVIC.BOSNIAN
IE.SLAVIC.SLOVENIAN
IE.ROMANCE.VLACH
IE.SLAVIC.POLISH
IE.SLAVIC.MACEDONIAN
IE.SLAVIC.SLOVAK
IE.SLAVIC.RUSSIAN
IE.SLAVIC.BULGARIAN
IE.SLAVIC.BELARUSIAN
IE.SLAVIC.SERBOCROATIAN
IE.SLAVIC.CROATIAN
IE.SLAVIC.NINILCHIK_RUSSIAN
IE.SLAVIC.LOWER_SORBIAN
IE.SLAVIC.KASHUBIAN
IE.SLAVIC.UKRAINIAN

E Concept Tree "Mountain" of the Indo-European languages




IE.INDIC.ATTOCK
IE.IRANIAN.KARAK_PASHTO
IE.GERMANIC.ICELANDIC
IE.IRANIAN.BANNU_PASHTO
IE.IRANIAN.MUNJANI
IE.IRANIAN.MIRAN_SHAH_PASHTO
IE.IRANIAN.ORMURI
IE.IRANIAN.PARACHINAR_PASHTO
IE.INDIC.WESTERN_PANJABI_SHAHPUR
IE.INDIC.NEPALI
IE.INDIC.BENGALI
IE.INDIC.URDU
IE.INDIC.RAJBANSHI
IE.INDIC.SADRI
IE.INDIC.MARATHI
IE.INDIC.HINDI
IE.IRANIAN.KANDAHAR_PASHTO
IE.IRANIAN.OGHI_PASHTO
IE.INDIC.ORIYA
IE.INDIC.ASSAMESE
IE.INDIC.KONKANI
IE.INDIC.GUJARATI
IE.INDIC.GURU_KALASHA
IE.INDIC.MALDIVIAN
IE.ARMENIAN.WESTERN_ARMENIAN
IE.INDIC.SEPECIDES_ROMANI
IE.INDIC.SHEKHANI
IE.GERMANIC.PLAUTDIETSCH
IE.IRANIAN.JALLOZAI_PASHTO
IE.INDIC.PURIGAL_PHALURA
IE.GERMANIC.DANISH
IE.IRANIAN.DIR_PASHTO
IE.INDIC.BALAKDT_HINDKO
IE.INDIC.KALKDTI
IE.GERMANIC.FRISIAN_WESTERN
IE.GERMANIC.DUTCH
IE.GERMANIC.BRABANTIC
IE.INDIC.PAKHA_GOLAM_HINDKO
IE.GERMANIC.ALSATIAN
IE.GERMANIC.SWEDISH
IE.INDIC.KOHAT_CITY_HINDKO
IE.INDIC.KRAKAL_KALASHA
IE.INDIC.NORTHERN_AZAD_KASHMIR_GUJARI
IE.GERMANIC.ZEEUWS
IE.GERMANIC.LIMBURGISH
IE.GERMANIC.AFRIKAANS
IE.GERMANIC.WESTVLAAMS
IE.GERMANIC.STANDARD_GERMAN
IE.GERMANIC.FRANS_VLAAMS
IE.GERMANIC.NORTH_FRISIAN_AMRUM
IE.GERMANIC.STELLINGWERFS
IE.GERMANIC.NORTHERN_LOW_SAXON
IE.INDIC.NORTH_RUSSIAN_ROMANI
IE.INDIC.LITHUANIAN_ROMANI
IE.INDIC.LATVIAN_ROMANI
IE.GERMANIC.YIDDISH_WESTERN
IE.GERMANIC.YIDDISH_EASTERN
IE.GERMANIC.NORWEGIAN_BOKMAAL
IE.GERMANIC.BERNESE_GERMAN
IE.GERMANIC.LUXEMBOURGISH
IE.INDIC.MENDHAR_GUJARI
IE.GERMANIC.CIMBRIAN
IE.GERMANIC.SAXON_UPPER
IE.GERMANIC.EASTERN_FRISIAN
IE.GERMANIC.SWABIAN
IE.INDIC.PARGAM_NISAR_KHOWAR
IE.INDIC.PUNJABI_MAJHI
IE.INDIC.SERAIKI
IE.INDIC.FINNISH_ROMANI
IE.GERMANIC.JAMTLANDIC
IE.IRANIAN.MADYAN_PASHTO
IE.IRANIAN.MARDAN_PASHTO
IE.INDIC.LAMANI
IE.IRANIAN.WANECI
IE.GERMANIC.FAROESE
IE.IRANIAN.BATAGRAM_PASHTO
IE.IRANIAN.TIRAH_AFRIDI_PASHTO
IE.CELTIC.GAELIC_SCOTTISH
IE.IRANIAN.PASHIN_KAKARI_PASHTO
IE.IRANIAN.PESHAWAR_PASHTO
IE.IRANIAN.PISHIN_PASHTO
IE.IRANIAN.QUETTA_PASHTO
IE.IRANIAN.SHAHMIRZADI
IE.IRANIAN.SHUGHNI
IE.IRANIAN.SORKHEI
IE.IRANIAN.THAL_PASHTO
IE.INDIC.SINTI_ROMANI
IE.INDIC.VLAX_ROMANI
IE.INDIC.MACEDONIAN_DZAMBAZI_ROMANIAN
IE.INDIC.GURBET_ROMANI
IE.INDIC.LOVARA_ROMANI
IE.SLAVIC.SERBOCROATIAN
IE.SLAVIC.MACEDONIAN
IE.SLAVIC.CZECH
IE.SLAVIC.CROATIAN
IE.SLAVIC.BULGARIAN
IE.INDIC.KOSOVO_ARLI_ROMANI
IE.INDIC.BANATISKI_GURBET_ROMANI
IE.INDIC.MACEDONIAN_ARLI_ROMANI
IE.INDIC.SREMSKI_GURBET_ROMANI
IE.INDIC.SOFIA_ERLI_ROMANI
IE.INDIC.BUGURDZI_ROMANI
IE.IRANIAN.WANA_PASHTO
IE.INDIC.BHOJPURI
IE.INDIC.TALAGANG
IE.INDIC.DOLENJSKI_ROMANI
IE.INDIC.HUNGARIAN_VEND_ROMANI
IE.INDIC.BURGENLAND_ROMANI
IE.INDIC.SAVI
IE.INDIC.SELICE_ROMANI
IE.INDIC.ROMUNGRO_ROMANI
IE.INDIC.GURVARI_ROMANI
IE.INDIC.EAST_SLOVAK_ROMANI
IE.INDIC.SOUTHERN_HAZARA_GUJARI
IE.INDIC.SINGO_DI_GARI
IE.INDIC.SHERPUR_HINDKO
IE.INDIC.BROK-SKAD
IE.ARMENIAN.EASTERN_ARMENIAN
IE.INDIC.MANSEHRA_HINDKO
IE.IRANIAN.NORTHERN_PASHTO
IE.IRANIAN.VAFSI
IE.IRANIAN.IRON_OSSETIAN
IE.IRANIAN.DIGOR_OSSETIAN
IE.IRANIAN.MINGORA_PASHTO
IE.INDIC.SETTLED_SWAT_GUJARI
IE.GREEK.GREEK
IE.INDIC.PAKHA_GOLAM
IE.SLAVIC.SLOVENIAN
IE.SLAVIC.RUSSIAN
IE.SLAVIC.NINILCHIK_RUSSIAN
IE.SLAVIC.LOWER_SORBIAN_2
IE.SLAVIC.LOWER_SORBIAN
IE.SLAVIC.POLISH
IE.IRANIAN.CHERAT_PASHTO
IE.IRANIAN.CHAMAN_PASHTO
IE.IRANIAN.JUDEO_TAT
IE.INDIC.KESU_KHOWAR
IE.SLAVIC.BOSNIAN
IE.SLAVIC.UKRAINIAN
IE.SLAVIC.SLOVAK
IE.SLAVIC.KASHUBIAN
IE.SLAVIC.UPPER_SORBIAN

F Pseudocode for replacing missing entries

Algorithm 1 Finding missing concepts

```
Require: sample ASJP matrix
  MissingConceptList ← empty
  for language in sample ASJP matrix do
    for concepts in languages do
      if concept = 0 then                          ▷ check if concept is missing
        MissingConceptList add (concept, language)
      end if
    end for
  end for

  return MissingConceptList
```

Algorithm 2 Finding closest related languages and their concept representation

```
Require: expert tree
Require: MissingConceptList                          ▷ List produced in Algorithm 1
  RelatedLanguageList ← empty
  for Pair in MissingConceptList do
    find language of pair in expert tree
    if sister of language is leaf then
      if sister has concept representation then
        RelatedLanguageList add (concept representation, sister)
      else
        while no concept representation is found do
          go one node up
          if sister is leaf then
            if sister has concept representation then
              RelatedLanguageList add (concept representation, sister)
            end if
          else                          ▷ the sister is no leaf, but has more descendants
            if descendant are leaves then
              intermediateList ← empty
              for leaves in descendants do
                if leave has a representation then
                  intermediateList add (concept representation, sister)
                end if
              end for
              RelatedLanguageList add intermediateList
            end if
          end if
        end while
      end if
    else                          ▷ the sister node is no leaf, but has more descendants
      for descendants in sister do
        if descendants are leaves then
          intermediateList ← empty
          for leaves in descendants do
            if leave has a representation then
              intermediateList add (concept representation, sister)
            end if
          end for
          RelatedLanguageList add intermediateList
        end if
      end for
      while no concept representation is found do
        go one node up
        for descendants in upper node do
          intermediateList ← empty
          for leaves in descendants do
            if leave has a representation then
              intermediateList add (concept representation, sister)
            end if
          end for
          RelatedLanguageList add intermediateList
        end for
      end while
    end if
  end for

  return RelatedLanguageList
```

Algorithm 3 Finding the best concepts for the replacement

```
Require: RelatedLanguageList                          ▷ List produced in Algorithm 2
  ReplaceConceptList ← empty
  for Pairs in RelatedLanguageList do
    if language of Pair has only one closest related language then
      ReplaceConceptList add (concept, closes related language)
    else                                                ▷ multiple donor languages
      distance ← 0
      intermediateList ← empty
      for donor languages in Pairs do
        compute distance between donor language and target language    ▷ Use procedure
        described in Jäger (2013)
        if computed distance < distance then
          distance ← computed distance
          intermediateList ← (concept, donor language)
        end if
        ReplaceConceptList add intermediateList
      end for
    end if
  end for

  return RelatedConeptList
```

Algorithm 4 Replace the concepts

```
Require: RelatecConeptList                            ▷ List produced in Algorithm 3
  for Pair in RelatedConceptList do
    for languages in sample ASJP matrix do
      if target language = language from sample ASJP matrix then
        concept in language from sample ASJP matrix ← concept from Pair
      end if
    end for
  end for

  return updated matrix
```

G Pseudocode for calculating reticulations

Algorithm 5 Calculating reticulations

```
Require: language tree
Require: languages under distance threshold
Require: concept tree
  Reticulations ← empty                          ▷ create empty to store the information
  for language in languages under distance threshold do
    find mother node in language Tree
    Reticulations add (language, mother node)
  end for
  for pair in list do                            ▷ get eacht pair from the list created above
    get sister node of language in concept tree
    if sister is leaf then
      pair ← language, mother, sister node
    else                                         ▷ i.e.  sister is an inner node
      get all descendants
      pair ← language, mother, listofdescendants
    end if
  end for
  FinalReticulations ← empty
  for retic in Reticulations do
    if retic is language then
      FinalReticulations add retic
    else                                         ▷ i.e.  we have listofdescendants
      find smallest common ancestor of all languages in listofdescendants
      FinalReticulations add (language, mother, smallest common ancestor)
    end if
  end for

  return FinalReticulations   ▷ List of language and the two nodes to which it has
  reticulations
```

H A list of languages causing evolutionary events

Concept	Language	Concept Representation
"I"	ROMANIAN	*ew*
"you"	NORWEGIAN_BOKMAAL	*d3*
"you"	FAROESE	*ti*
"one"	PORTUGUESE	*u*
"two"	DANISH	*to7*
"person"	ENGLISH	*pers3n*
"dog"	ENGLISH	*dag*
"tree"	ROMANIAN	*pom-arbore*
"ear"	FRENCH	*ore*
"eye"	DUTCH	*oX*
"eye"	ARPITAN	*yi*
"eye"	FRENCH	*3y*
"nose"	ICELANDIC	*nev*
"tooth"	CATALAN	*den*
"knee"	SPANISH	*rodiya*
"star"	ROMANIAN	*stea*
"fire"	STANDARD_GERMAN	*foia*
"path"	FRENCH	*rut*
"mountain"	ENGLISH	*maunt3n*
"night"	ICELANDIC	*nout*
"night"	CATALAN	*nit*
"new"	FAROESE	*nuigur*
"new"	STANDARD_GERMAN	*noi*

Table H.1: Languages causing evolutionary events between two language families

Concept	Language	Concept Representation
"you"	ICELANDIC	*8u*
"you"	DUTCH	*yEi-y3*
"we"	ARPITAN	*no-nu*
"we"	CATALAN	*nuzaltr3s*
"we"	FRENCH	*nu*
"we"	ITALIAN	*noi*
"we"	PORTUGUESE	*noS*
"we"	ROMANIAN	*noi*
"we"	SARDINIAN	*nos*
"we"	SPANISH	*nosotros*
"one"	DANISH	*e7n*
"one"	DUTCH	*en*
"one"	ENGLISH	*w3n*
"one"	FAROESE	*oin-oit*
"one"	FRISIAN_WESTERN	*i3n-yin*
"one"	STANDARD_GERMAN	*ains*
"two"	ENGLISH	*tu*
"two"	NORWEGIAN_BOKMAAL	*tu*
"two"	SWEDISH	*to*
"person"	DUTCH	*pErson-mEns*
"person"	ARPITAN	*omu*
"person"	FRENCH	*om*
"fish"	DUTCH	*vis*
"fish"	FRISIAN_WESTERN	*fisk*
"fish"	STANDARD_GERMAN	*fiS*
"fish"	CATALAN	*peS*
"fish"	SPANISH	*peskado-pes*
"louse"	ARPITAN	*pu*
"louse"	FRENCH	*pu*
"tree"	DUTCH	*bom*
"tree"	FRISIAN_WESTERN	*bi3m-bEm*
"tree"	STANDARD_GERMAN	*baum*
"tree"	SPANISH	*arbol-palo*
"leaf"	DANISH	*blE8*
"leaf"	FAROESE	*bla*
"leaf"	ICELANDIC	*pla8*
"leaf"	NORWEGIAN_BOKMAAL	*blod*
"leaf"	SWEDISH	*lev*
"skin"	DANISH	*hu87*
"skin"	ENGLISH	*skin*
"skin"	FAROESE	*Sid*
"skin"	ICELANDIC	*hu8-sTin*
"blood"	DANISH	*blo87*
"blood"	DUTCH	*blut*
"blood"	ENGLISH	*bl3d*
"blood"	FAROESE	*ble*
"blood"	FRISIAN_WESTERN	*blu3t*
"blood"	ICELANDIC	*plou8*

Concept	Language	Concept Representation
"blood"	NORWEGIAN_BOKMAAL	*blud*
"blood"	STANDARD_GERMAN	*blut*
"blood"	SWEDISH	*bud*
"bone	DUTCH	*ben*
"bone	FRISIAN_WESTERN	*boNk3*
"bone"	STANDARD_GERMAN	*knoX3n*
"bone"	ROMANIAN	*os*
"bone"	DUTCH	*ben*
"bone"	FRISIAN_WESTERN	*boNk3*
"bone"	STANDARD_GERMAN	*knoX3n*
"bone"	ROMANIAN	*os*
"horn"	FRISIAN_WESTERN	*ho3n-han*
"horn"	ROMANIAN	*korn*
"nose"	DUTCH	*nes*
"nose"	FRISIAN_WESTERN	*nes-nas3*
"nose"	ARPITAN	*na-no*
"nose"	FRENCH	*ne*
"tooth"	DANISH	*tEn7*
"tooth"	DUTCH	*tant*
"tooth"	ARPITAN	*dE*
"tooth"	FRENCH	*da*
"tongue"	DANISH	*toN3*
"tongue"	DUTCH	*toN*
"tongue"	ENGLISH	*t3N*
"tongue"	FRISIAN_WESTERN	*toN3-toNg3*
"tongue"	STANDARD_GERMAN	*cuN3*
"hand"	DANISH	*han7*
"hand"	DUTCH	*hant*
"hand"	ENGLISH	*hEnd*
"hand"	FAROESE	*hond*
"hand"	FRISIAN_WESTERN	*hon*
"hand"	ICELANDIC	*hEnt*
"hand"	NORWEGIAN_BOKMAAL	*hond*
"hand"	STANDARD_GERMAN	*hant*
"hand"	SWEDISH	*hEn*
"hand"	ITALIAN	*pEtto*
"hand"	ROMANIAN	*pept-s3n*
"liver"	DANISH	*lea*
"liver"	DUTCH	*lev3r*
"liver"	ENGLISH	*liv3r*
"liver"	FRISIAN_WESTERN	*lew3r*
"liver"	STANDARD_GERMAN	*leb3r*
"liver"	DANISH	*lea*
"drink"	DANISH	*drEg3*
"drink"	ENGLISH	*drink*
"see"	DANISH	*se7*
"see"	ENGLISH	*si*
"see"	FAROESE	*suiga*

Concept	Language	Concept Representation
"see"	ICELANDIC	*sau*
"hear"	ARPITAN	*ekota-exota*
"hear"	CATALAN	*s3nti*
"hear"	FRENCH	*otadr*
"hear"	SARDINIAN	*intendere*
"die"	DANISH	*de7*
"die"	ENGLISH	*dEi*
"die"	NORWEGIAN_BOKMAAL	*de*
"die"	SWEDISH	*de*
"come"	DANISH	*kam3*
"come"	FRISIAN_WESTERN	*kom3*
"come"	NORWEGIAN_BOKMAAL	*kom3*
"come"	SPANISH	*veni*
"sun"	DANISH	*sol*
"sun"	FAROESE	*sel*
"sun"	ICELANDIC	*soul*
"sun"	NORWEGIAN_BOKMAAL	*sol*
"sun"	SWEDISH	*sul*
"sun"	CATALAN	*sol*
"sun"	PORTUGUESE	*sol*
"sun"	ROMANIAN	*soare*
"sun"	SPANISH	*sol*
"water"	ARPITAN	*Ewa-Ega*
"water"	CATALAN	*aix3*
"stone"	DANISH	*sten*
"stone"	DUTCH	*sten*
"stone"	ENGLISH	*ston*
"stone"	FRISIAN_WESTERN	*sti3n*
"stone"	NORWEGIAN_BOKMAAL	*stEin*
"stone"	STANDARD_GERMAN	*Stain*
"stone"	SWEDISH	*sten*
"fire"	DUTCH	*vir*
"fire"	ENGLISH	*fEir*
"fire"	FRISIAN_WESTERN	*fu3r*
"path"	ARPITAN	*Sami-cami*
"path"	CATALAN	*k3mi*
"path"	ROMANIAN	*k3rare*
"night"	ARPITAN	*nE-nE*
"night"	FRENCH	*nui*
"full"	ENGLISH	*ful*
"full"	FAROESE	*fudur*
"full"	ICELANDIC	*fitlir*
"ful"	ROMANIAN	*plin*
"name"	ARPITAN	*no*
"name"	CATALAN	*nom*
"name"	FRENCH	*no*

Table H.2: Languages causing evolutionary events within a language family