Algorithmic advancements in Computational Historical Linguistics

Dissertation
zur
Erlangung des akademischen Grades
Doktor der Philosophie
in der Philosophischen Fakultät

der Eberhard Karls Universität Tübingen

vorgelegt von

Johannes Wahle

aus

Soest

2021

Gedruckt mit Genehmigung der Philosophischen Fakultät
der Eberhard Karls Universität Tübingen

Dekan: Prof. Dr. Jürgen Leonhardt

Hauptberichterstatter: Prof. Dr. Gerhard Jäger
Mitberichterstatterin: Prof. Dr. Wiebke Petersen

Tag der mündlichen Prüfung: 12. März 2021

Universitätsbibliothek Tübingen, TOBIAS-lib

Algorithmic advancements in Computational Historical Linguistics

Abstract

The use of computational methods in historical linguistics has seen a large boost in recent years. An increasing availability of machine readable data and the growing power of computers fostered this development. While the computational methods which are used in this research stem from different scientific disciplines, a lot of tools from computational biology have found their way into this research. Drawing inspiration from advancements in related fields, this thesis aims at improving existing computational methods in different disciplines of computational historical linguistics.

Using advancements from machine learning and natural language processing research, I present an updated training regime for cognate detection algorithms. Besides achieving state of the art performance in a cognate clustering task, the updated training scheme considerably improved computation time.

Following up on these results, I develop a novel combination of tools from bioinformatics and historical linguistics is developed. By defining an explicit model of sound evolution, I include the notion of evolutionary time into a cognate detection task. The resulting posterior distributions are used to evaluate the model on a standard cognate detection task.

A standard problem in phylogenetic research is the inference of a tree. Current quasi "industry-standard" methods use the classical Metropolis-Hastings algorithm. However, this algorithm is known to be rather inefficient for high dimensional and correlated data. To solve this problem, I present an algorithm which uses Hamiltonian dynamics in the last chapter.

Algorithmic advancements in Computational Historical Linguistics

Zusammenfassung

Computergestützte Methoden in der historischen Linguistik haben in den letzten Jahren einen großen Aufschwung erlebt. Die wachsende Verfügbarkeit maschinenlesbarer Daten förderten diese Entwicklung ebenso wie die zunehmende Leistungsfähigkeit von Computern. Die in dieser Forschung verwendeten Berechnungsmethoden stammen aus verschiedenen wissenschaftlichen Disziplinen, wobei Methoden aus der Bioinformatik sicherlich die Initialzündung gaben. Diese Arbeit, die sich von Fortschritten in angrenzenden Gebieten inspirieren lässt, zielt darauf ab, die bestehenden Berechnungsmethoden in verschiedenen Bereichen der computergestützten historischen Linguistik zu verbessern.

Mit Hilfe von Fortschritten aus der Forschung aus dem maschinellen Lernen und der Computerlinguistik wird hier eine neue Trainingsmethode für Algorithmen zur Kognatenerkennung vorgestellt. Diese Methode erreicht an vielen Stellen die besten Ergebnisse im Bereich der Kognatenerkennung. Außerdem kann das neue Trainingsschema die Rechenzeit erheblich verbessern.

Ausgehend von diesen Ergebnissen wird eine neue Kombination von Methoden der Bioinformatik und der historischen Linguistik entwickelt. Durch die Definition eines expliziten Modells der Lautevolution wird der Begriff der evolutionären Zeit in die Kognatenerkennung mit einbezogen. Die sich daraus ergebenden posterioren Verteilungen werden verwendet, um das Modell anhand einer standardmäßigen Kognatenerkennung zu evaluieren.

Eine weitere klassische Problemstellung in der pyhlogenetischen Forschung ist die Inferenz eines Baumes. Aktuelle Methoden, die den "quasi-industriestandard" bilden, verwenden den klassischen Metropolis-Hastings-Algorithmus. Allerdings ist bekannt, dass dieser Algorithmus für hochdimensionale und korrelierte Daten vergleichsweise ineffizient ist. Um dieses Problem zu beheben, wird im letzten Kapitel ein Algorithmus vorgestellt, der die Hamilton'sche Dynamik verwendet.

Contents

1 Introduction 1

2 (Computational) Historical Linguistics 4
2.1 The Comparative Method 5
2.1.1 The Comparative Method — narrow sense 6
2.2 The Computational Turn 9
2.2.1 Databases 10
2.2.2 Computational Approaches 12

3 Phylogenetic Inference in Computational Biology 14
3.1 Background on Markov Models 15
3.1.1 Markov Chains 15
3.1.2 Hidden Markov Models 17
3.2 Background on Alignment 24
3.2.1 Score Based Alignment 26
3.2.2 Pair Hidden Markov Models 31
3.2.3 Statistical Alignment 36
3.3 Models of DNA Evolution 44
3.3.1 The Jukes-Cantor Model 46
3.3.2 The General Time Reversible Model 47
3.4 Trees 49
3.5 Felsenstein's Algorithm and the Pulley Principle 52

4 Automatic Cognate Identification and Phylogenetic Inference 55
4.1 Sequence Alignment and Cognate Detection in the Literature 57
4.2 Phylogenetic Inference in the Literature 60
4.3 Automatic and Unsupervised Cognate Identification 62
4.3.1 Models 63
4.3.2 Online EM 68
4.3.3 Clustering Algorithm 69
4.3.4 Experiments 70
4.3.5 Comparing Systems 73
4.3.6 Results 75
4.3.7 Discussion 80
4.3.8 Conclusion 82

4.4 Phylogenetic Inference from automatic cognate judgments 83
4.4.1 Two Additional Approaches 83
4.4.2 Bayesian Phylogenetic Inference 86
4.4.3 Experiments 87
4.4.4 Results 89

5 **Cognacy Estimation under a Time-Dependent Alignment Model** **90**
5.1 Q-matrices for Sounds 91
5.2 Building the Model 95
5.3 Parameter Estimation 99
5.3.1 Improved Sampling 102
5.4 Cognacy Estimation in the Time-Dependent Model 104
5.5 Experiments 107
5.6 Results 107
5.7 Discussion 109
5.7.1 Comments 111
5.8 Conclusion 112

6 **No-U-Turn Sampling for Phylogenetic Trees** **113**
6.1 Hamiltonian Dynamics and the No-U-Turn Sampler 114
6.1.1 Leapfrog approximation 115
6.1.2 No-U-Turn sampler 117
6.2 Hamiltonian Dynamics in the Tree Space 119
6.3 No-U-Turn Sampling in the Tree Space 122
6.4 Empirical Evaluation 124
6.4.1 Computational Performance 126
6.5 Conclusion 128

7 **Conclusion** **129**
7.1 Outlook and Future Work 131
7.2 Final Remarks 131

References **133**

Listing of figures

3.1 Markov chain for DNA sequences 17
3.2 Example of an alignment graph 27
3.3 FSA and probabilistic model of affine gap alignment 31
3.4 Pair HMM (cf. Durbin et al., 2001, p. 82) 32
3.5 Random Model (cf. Durbin et al., 2001, p. 83) 34
3.6 Sequence of links 37
3.7 Complete history of fates 40
3.8 HMM formulation of the TKF91 process 41
3.9 HMM formulation of the TKF92 process 42
3.10 Evolutionary process of nucleic acids (symbolic) 45
3.11 Different models of DNA evolution 45
3.12 Example for a rooted tree with branch lengths 52

4.1 Example of a sigmoid function 64
4.2 PHMM as proposed by Mackay and Kondrak (2005) 67
4.3 Plots of m and α against B-cubed F-scores for out-of-family training . 78
4.4 Distribution of the average difference of word lengths across concepts 81
4.5 Quartet arrangements 88

5.1 Illustration of the slice sample 103
5.2 B-cubed F-scores from alignment scores 108
5.3 B-cubed F-scores from probabilities 110

6.1 Example of a NNI move 121
6.2 Distribution of GQD scores 126
6.3 Shrink factor against time 127

Listing of tables

2.1 Example data from Weiss (2015) 8

3.1 Number of alignments 25
3.2 Parametrization of the general time reversible model 48

4.1 Example of a word list for five languages 56
4.2 Characteristics of the data 72
4.3 Comparison of B-cuped F-scores for different models 76
4.4 Best settings of m and α for Online variants of PMI and PHMM . . . 77
4.5 Results of within family training 78
4.6 B-cubed F-scores for different cognate detection methods across the language families 85
4.7 Results of the phylogenetic inference algorithms 89

5.1 Dolgopolsky's sound classes 93
5.2 Alignment scores 106
5.3 Median B-cubed F-scores and quantile per setting 108
5.4 Highest B-cubed F-scores for cut-off values and quantile per setting . 109

6.1 Overview of the data sets used to test p-NUTS 124
6.2 Results from the MCMC runs for MrBayes and p-NUTS 125

To Angelina and my parents. Thank you for supporting me.

Danksagung

Diese Arbeit wäre ohne die Hilfe einiger Menschen nicht möglich gewesen. Zu aller erst möchte ich meinem Betreuer Gerhard Jäger dafür danken, dass er mir dieses spannende Themenfeld eröffnet hat. Er hat mir die Freiheit gelassen, meine Stärken zu finden und meine eigenen Fehler zu begehen. Trotzdem hat er das Ziel nie aus den Augen verloren und es geschafft, mich hin und wieder auf den richtigen Pfad zu lenken. Ohne seine Unterstützung und Anleitung wäre ich wahrscheinlich immer noch im Dschungel von Ideen und Ansätzen unterwegs.

Besonders möchte ich auch Marisa Köllner danken, die mir geholfen hat den Kopf über Wasser zu halten. Sie hat es immer als erstes mitbekommen, wenn etwas nicht nach Plan gelaufen ist und trotzdem nicht das Büro tauschen wollte. Ich werde immer gerne an unsere gemeinsame Arbeitszeit zurückdenken. Taraka Rama, Johannes Dellert, Tino Sehring, Christian Bentz und Igor Yannovich möchte ich für ihr hilfreiches Feedback danken. Auch wenn ich es zuerst nicht wahrhaben wollte, ihr hattet meistens recht. Ohne eure Gesellschaft und die Diskussionen mit euch wäre diese Arbeit nicht möglich gewesen. Ich möchte auch allen anderen Projektmitarbeiterinnen- und mitarbeitern in EVOLAEMP danken, die ich hier nicht alle aufzählen kann. Liebe Dani, ohne deine Hilfe könnte niemand diese Dissertation vernünftig lesen, vielen Dank für das unendliche Korrigieren.

Diese Arbeit bildet den Abschluss meiner Ausbildungszeit in Tübingen. In dieser Zeit hatte ich das überaus große Glück, Menschen zu treffen, die mir sehr ans Herz gewachsen sind. Ich möchte Peter und Steffen für unsere gemeinsame Zeit, den Spaß und auch die Unterstützung in schwierigen Momenten danken. Ihr habt es geschafft, dass diese Zeit zu einer der schönsten meines Lebens wurde. Danke Christl, für dein offenes Ohr in allen Lebenslagen und den Kaffee.

Ich habe das unglaubliche Glück eine tolle Familie zu haben, die es mir ermöglicht hat meinen Weg zu gehen. Vielen Dank an meine Eltern, die mich ermutigt haben und für mich da sind. Vieles von dem, was ich von euch gelernt habe, habe ich lange nicht zu schätzen gewusst, bis es mir durch so manche Widrigkeit geholfen hat.

Der letzte und größte Dank geht an Angelina. Vielen Dank, dass du für mich da warst, als es hart war und es kein Licht am Ende des Tunnels gab. Danke, dass du diese emotionale Achterbahnfahrt mit mir durchgemacht hast, obwohl du kein Ticket für diese Fahrt gelöst hast. Danke für deine Unterstützung als es schien, es würde kein Ende nehmen. Ohne Dich wäre diese Arbeit nie zustande gekommen.

1 Introduction

We always want to find the origin of something or the reason why something is happening. This curiosity has been a driving force of scientific endeavors. Human language is no exception to this. Thus, it is not surprising that the history and the origin of languages has also been a major point of interest. Shortly before Darwin presented his ideas on evolution and also noted the similarities between the evolution of species and languages (Darwin, 1871), the German linguist August Schleicher published a tree displaying relationships between the Indo-European languages as well as attempting to reconstruct the Proto-Indo-European language (Schleicher, 1861). After becoming aware of Darwins work, Schleicher drew several parallels between the evolution of languages and species and advocated rigorous treatment of the study of language evolution with the tools of the natural sciences (Schleicher, 1863). As Konrad Koerner puts it in his foreword to Schleicher's *Die Sprachen Europas in systematischer Übersicht: linguistis-*

che Untersuchungen: “For him [Schleicher, authors note], languages could be analyzed like organisms [...]” (p. XLVIII Schleicher, 1983).

With the increasing availability of machine-readable linguistic data and the advancement of computational methods in recent years, these parallels gave rise to the application of algorithms and methods from computational biology in a linguistic setting. Atkinson and Gray (2005) and Croft (2001), building on Schleicher’s work, present a systematic overview of similarities in data types and mechanisms of linguistic and biological evolution. These similarities have paved the way for methods from computational biology into linguistic research. The identification of groups of words which are related to each other via a common ancestor has always been an important aspect of this kind of research. While this process previously relied heavily on manual work, computational methods proved to be useful for this task. Furthermore, cognate word pairs are important for phylogenetic inference. Modern algorithms from computational biology are able to build fairly accurate trees of language families purely based on cognacy data. Thus, cognates play an important role in this area of research.

The aim of this thesis is to introduce new techniques to the realm of computational historical linguistics. Thus, it tries to improve existing models of cognate identification by using Markov Models. Such models are a stochastic framework which can be used to model sequential data. They were already successfully used in a number of applications from computational biology to computational linguistics. Thus, they are well studied and provide a powerful framework. In a second attempt, this thesis tries to bring together the task of cognate identification and phylogenetic inference. In traditional historical linguistics, establishing cognate word pairs and inferring language family trees went hand in hand. Modern approaches which bring together computational biology and historical linguistics fall short in this regard. In this thesis, I bring together cognate identification and phylogenetic inference by using an evolutionary model. This model makes explicit assumptions about processes which underlie the evolutionary processes.

Finally, this thesis proposes a method which improves the algorithms used for phylogenetic inference. All algorithms in this area are based on Markov Chain Monte Carlo methods. The traditional algorithms for phylogenetic inference use a variant which is known to be rather ineffective. Building on recent advancements, this thesis aims to make these algorithms more effective.

Chapters 1 and 2 of this thesis introduce the relevant concepts in historical linguistics and computational biology. Chapter 3 shows how current algorithms used for the identification of cognates can be improved. Furthermore, these cognates are used in the downstream application of inferring phylogenetic trees from inferred cognate sets. In the chapter 4, I develop a model which tries to unite cognate identification and phylogenetic inference in one evolutionary model. The idea is to derive an explicit model of sequence evolution for linguistics which can be used to infer cognacy and phylogenetic relatedness. Chapter 5 is dedicated to the discussion of recent advancements in Markov Chain Monte Carlo techniques and their application for computational historical linguistics. I conclude this thesis in chapter 6 with a discussion of the main results and a short outlook on further topics and possible further research.

2 (Computational) Historical Linguistics

Historical linguistics has a very long standing tradition. Taking a diachronic look at languages started at least as early as the 1820s with the work of Jacob and Wilhelm Grimm. Thus, it is arguably the oldest sub-discipline of linguistics. Research in historical linguistics has shed light on the principles governing language change as well as upon the societies which spoke these languages. Such findings also proved to be valuable for other fields dealing with prehistory (Renfrew, 1987; Haak et al., 2015, see also the references in Jäger (2018)). A technique which is central to historical linguistics is the *comparative method* which according to Weiss (2015) is used as such already since Schleicher (1852). The different parts integral to the comparative method (see section 2.1) are in itself very algorithmic in nature. This aspect, among other reasons, has lead to a computational turn in the area of historical linguistics. The second main part of this chapter illustrates algorithmic approaches to the study of language history. This research has ties to computational biology (some relevant methods are introduced in chapter 3 in the area of *computational phylogenetic linguistics*, computational linguistics (see for example chapter 4) and several other disciplines.

2.1 The Comparative Method

The comparative method is one of the cornerstones of classical historical linguistics. In its "narrow sense" (Ross and Durie, 1996, p. 3), the comparative method describes a series of steps necessary to systematically reconstruct (parts of) an ancestral language from its reflexes in daughter languages. In its "wider sense" (Ross and Durie, 1996, p. 3), the comparative method is conjoined with the theory of the "Neogrammarian hypothesis" (Ross and Durie, 1996). One of the main parts of the neogrammarian hypothesis is that there are no exceptions when it comes to sound change or as Osthoff and Brugmann put it in the original:

> Aller lautwandel, so weit er mechanisch vor sich geht, vollzieht sich nach ausnahmslosen gesetzen, d. h. die richtung der lautbewegung ist bei allen angehörigen einer sprachgenossenschaft, ausser dem fall, dass dialektspaltung eintritt, stets dieselbe, und alle wörter, in denen der der lautbewegung unterworfene laut unter gleichen verhältnissen erscheint, werden ohne ausnahme von der änderung ergriffen (Osthoff and Brugmann, 1878, p. 8).[1]

The neogrammarian hypothesis aims to ground the science of language history and their tools onto firm ground in order to have a scientific method which can be rigorously analyzed. While this hypothesis makes pretty strong assumptions, it has been very vital

[1] The spelling is as in the original.
This translation is due to Lehmann (1967): "[E]very sound change, inasmuch as it occurs mechanically, takes place according to laws that admit no exception. That is, the direction of the sound shift is always the same for all the members of a linguistic community except where a split into dialects occurs; and all words in which the sound subjected to the change appears in the same relationship are affected by the change without exception."
In his introduction to August Schleicher's *Die Sprachen Europas ins systematischer Übersicht: linguistische Untersuchungen*, Koerner argues that Osthoff's view can already be found in Schleicher's work (Schleicher, 1983, p. LI).

to (historical) linguistics and lead to important findings about the history of languages. The comparative method in the wider sense is concerned with questions regarding the implications of the hypothesis and how to deal with cases that seem to violate it, e.g., what are probable sociolinguistic factors, cognitive or physiological factors influencing language change? (See Ross and Durie (1996) for an overview.) The focus of this introduction will be on the comparative method in its narrow sense since it lends itself readily to automation. Scholars have used algorithmic approaches to automatize several parts of the comparative method (see section 2.2).

2.1.1 The Comparative Method — narrow sense

Ross and Durie (1996) summarize the comparative method in its narrow sense into a sequence of seven instructions. This set of instructions describes a process which ideally leads to a systematic assessment of the structure of relatedness between languages of a given group.[2]

1. Based on some evidence, select a set of languages which are presumably genetically related, i.e., they form a family.
2. Collect a set of alleged cognates for this family.
3. Identify sound correspondences within the cognate sets.
4. Reconstruct a protolanguage based on the sound correspondences.
5. Establish subgroups within the family based on shared innovations.
6. Build a family tree based on the subgroups from the previous step.
7. Create an etymological dictionary listing the shared innovations and reconstructions.

[2]See Jäger and List (2016) and Jäger (2018) for a visualization of these steps.

Although these steps suggest that these steps should be performed sequentially, the comparative method often works in an iterative fashion. Steps two and three, for example are often iterated, such that new cognates are found based on sound correspondences detected previously. Another important point regarding the comparative method, which becomes evident from the formulation of the steps, is that it does not generate hypotheses about relatedness between languages but rather functions as a tool to validate or invalidate such hypotheses (Weiss, 2015).

An important term for the remainder of this thesis as well as the comparative method is the notion of a *cognate*. Two words are *cognates* if they are derived from the same word in a common ancestral language. For example, English *fish* and German *Fisch* are cognates, their common ancestor can be traced to the stage of Proto-Indo-European **fiska-* (Kroonen, 2013). Identifying such cognates is an important aspect of the comparative method as well as for computational methods in historical linguistics. The definition of the term *cognate* only requires that two words which are cognate are derived from a common ancestor. Consequently, the meaning of a word does not influence the notion of cognacy. The English word *bone* and the German word *Bein* (leg) are cognate although they have different meanings. Both are derived from the Proto-Indo-European word **baina-* which has the two different meanings 'bone' and 'leg' (Kroonen, 2013). Nichols notes "that any vocabulary set displaying the regular sound correspondences is in fact cognate, how-ever far-fetched the semantic correspondences" (Nichols, 1996, p. 41).

To illustrate the comparative method, consider the following example taken from Weiss (2015). As the data in table 2.1 (see p. 8) shows, there are correspondences of word initial sounds for (a) Spanish, Portuguese, Catalan and Italian *k* and Old French *tʃ* in the first group, (b) Spanish θ, Catalan and Portugese *s*, Italian *tʃ* and Old French *ts* in the second group, and (c) a corresponding *s* for all languages. After picking up these correspondences, the reconstruction of ancestral sounds can be done by consid-

Sp.	Po.	Cat.	OF.	It.	Gloss	Group
kampo	kãpu	kam	tʃamp	kampo	field	a
kanta	kanta	kantə	tʃantə	kanta	sings	a
θjelo	sɛw	sɛl	tsjel	tʃɛo	heaven	b
seko	seku	sɛk	sɛk	sekːo	dry	c
sako	saku	sak	sak	sakːo	sack	c

Table 2.1: Example data taken from Weiss (2015, p. 129). Sp. Spanish, Po. Portuguese, Cat. Catalan, OF. Old French, It. Italian.

ering the distributions of the different sounds. At close inspection, the first two sets reveal that the sounds in the second set occur before a front vowel whereas the first occurs before a non-front vowel. This complementary distribution of sounds allows the proposal of a proto-segment which developed into two different sounds depending on the environment. The third set does not stand in a complementary distribution to the other two. Thus, a second proto-segment needs to be assumed for the third set. The proto-segement which can be reconstructed for the third set is **s*. For the first proto-segment, there is no obvious choice. However, if Sardinian, another language related to the ones above, gets taken into account, the picture changes. The reflexes for Sardinian in the respective sets are *k*, *k* and *s*.[3] By taking these data points into account, it is possible to decide on **k* as the proto-sound, since it is the only reflex which is shared among the two sets. As Weiss (2015) illustrates, this process can be taken further by incorporating more data. Accommodating additional items leads to a more complete picture and allows then to propose a language tree which is supposed to be the second but last step of the comparative method.

It is important to note that deciding on the proto-segment **k* was only possible in the presence of the evidence provided by Sardinian. This case illustrates a crucial aspect of the comparative method, namely that reconstructions of proto-segments and identification of correspondence sets heavily depend on the presence of complete data. Therefore, the proper collection of data is important for a successful application of the com-

[3]The different words are in order: kampu, kanta, kelu, sikːu, sakːu (cf. Weiss, 2015)

parative method. Although presented as two separate steps so far, the identification of cognate sets and the detection of sound correspondences heavily overlap in practice. By proposing putative cognate sets, assumptions about sound correspondences are already implicit (Ross and Durie, 1996).

2.2 The Computational Turn

The computational turn in historical linguistics got serious traction with the availability of sufficient data sets which allowed the application of algorithmic methods to answer questions in historical linguistics. As elaborated above, the comparative method, a cornerstone of historical linguistics, has several aspects to it: the identification of cognates and sound correspondences, detection of sound laws and reconstruction of phylogenetic trees. The computational approach to historical linguistics treats these different aspects separately. Although these parts are very much interconnected, and usually inform each other, it currently does not seem feasible to fully implement all aspects of the comparative method. Jäger and List (2016) compare the classical comparative method to approaches in computational historical linguistics (CHL). They show that also in computational historical linguistics, there is an overlap between cognate identification and sound correspondence detection. In comparison to the traditional approach to these tasks, it is possible to circumvent implicit assumptions or make them an explicit parameter of the model in a computational approach.

Several methods which are used in CHL are adapted from computational biology. Chapter 3 of this thesis gives a short introduction to some important methods. Atkinson and Gray (2005) provide an overview of parallels between these different research areas. As mentioned above and already illustrated by Atkinson and Gray (2005), parallels between linguistic and biological species are already pointed out in Darwin's *The descent of Man* (Darwin, 1871) and August Schleicher's work (Schleicher, 1863). Atkinson and Gray note an important parallel between the characters which are used. DNA sequences are made of single discrete units. The same can arguably be said for sentences or words when they are transcribed. In both the biological and the linguistic case, there is a sequence of discrete symbols which can be analyzed. Another parallel which is also important for this thesis is between *homology* and *cognacy*. Although Fitch (2000) discusses problems in the proper usage of the term *homology* in the context of biology, the working definition of homology is: "The relationship of any two characters that have descended, usually with divergence, from a common ancestral character" (Fitch, 2000, p. 229). This definition is strikingly similar to the definition of the term *cognate* above. Both concepts describe the property of descending from a common ancestor. These and other parallels suggest that the methods from computational biology may be used in the area of CHL.

2.2.1 Databases

As mentioned above, the increasing availability and size of databases has fueled research in CHL. Generally, there are two types of databases in this research which can be distinguished. The main difference between the two databases is the presence of expert cognate judgments. Databases which are annotated for cognacy tend to focus on a particular language family. The other type of database focuses on covering a wide array of languages at the expense of cognacy annotation. Both types of databases aim to provide a list of translations of basic concepts for each language covered. These basic concept lists are termed *Swadesh lists*. Following Swadesh (1955) these concepts are supposed to be central to the language and therefore resistant towards borrowing.

The different databases are suited for different kinds of studies. While approaches to automatic cognate detection obviously need expert annotated cognate sets for evaluation purposes, the detection of sound correspondences greatly benefits from large amounts of data. In the area of phylogenetic inference, i.e., the use of computational methods in order to build family trees, there are two different approaches: the character

based method and the distance based method. The character based method utilizes cognate coded data to infer language trees using models of character evolution. Distance based approaches extract a distance measure on the basis of phonetic similarity to build a tree model.
The following list of databases does not aim for completeness but rather introduces well-known databases and ones which are used throughout the remainder.

Cognacy-coded Databases

Indo-European database

The Indo-European Lexical database (IELex) was created by Dyen et al. (1992) and curated by Michael Dunn.[4] The IELex database is not transcribed in uniform IPA and retains many forms transcribed in the Romanized IPA format of Dyen et al. (1992). It is a database which gives translations of 207 concepts in 52 languages.

Austronesian Basic Vocabulary Database

The Austronesian Basic Vocabulary Database (ABVD) (Greenhill et al., 2008) has word lists for 210 Swadesh concepts and more than 1500 languages.[5] It does therefore cover almost the entire language family. The database does not have transcriptions in a uniform IPA format.

Non Cognacy-coded Databases

Automated Similarity Judgment Program

The Automated Similarity Judgment Program (ASJP) is a massive cross-linguistic database which in version 18 (Wichmann et al., 2018) provides word lists for more than 5000

[4] `http://ielex.mpi.nl/`
[5] `https://abvd.shh.mpg.de/austronesian/`

languages.[6] In order to achieve such a high coverage of different languages, the ASJP database focuses on just a 40 item word list (Holman et al., 2008). All items in the ASJP database are phonetically transcribed using a 41 symbol alphabet designed to encode all commonly occurring sounds (Brown et al., 2008).

NorthEuraLex

The NorthEuraLex database (Dellert and Jäger, 2017; Dellert, 2015) is a database which contains word lists of 1016 items for 107 languages from Northern Eurasia and some important contact languages from neighboring families.[7] The words come in an IPA encoding which is automatically generated from the orthographic source. In comparison to other databases, NorthEuralex aims at a high coverage per language. The list of concepts for this database was automatically selected (Dellert and Buch, 2018).

2.2.2 Computational Approaches

Chapters 4 and 5 deal with some aspects of the automation of steps of cognate detection, identification of sound correspondences and phylogenetic reconstruction as well as the relevant related literature. In order to provide additional context, there are some studies which deserve to be presented and do not fit into the later parts.

Bouchard-Côté et al. (2013) propose a probabilistic method which automatizes the complex process of reconstructing proto-sounds. Given a collection of cognate sets from modern day languages, their approach uses a probabilistic string transducer which takes into account the context of each symbol. Their system was developed and tested on data from the ABVD database. Bouchard-Côté et al. (2013) are able to get about seven out of eight phonemes right in their reconstruction. They thus claim that their system is capable of a large scale reconstruction of proto-languages despite making considerable simplifying assumptions. Dellert (2017) mentions that the Austronesian lan-

[6] `https://asjp.clld.org/`
[7] `http://northeuralex.org/`

guage family is considered to be an easier case regarding the reconstruction and considers the task of automated reconstruction of proto-forms to still be an open problem.

Another aspect of reconstruction was studied by Jäger and List (2018) and List (2015). Instead of reconstructing actual word forms, their aim is to reconstruct the evolution of word forms in terms of cognate classes. Jäger and List (2018) explore the performance of different algorithms for reconstructing ancestral states from computational biology. Upon closer inspection, they observe that the cases where the algorithms fail in predicting the correct ancestral form are due to instances of other linguistic processes interfering with the process of evolution. List (2015) uses reconstructed cognate forms to paint a picture of lexical transfer in Chinese dialects.

Dellert (2017, 2016) uses causal inference algorithms to study the flow of lexical material between languages. Based on automatically inferred cognate sets, these methods show the transfer of lexical material between languages. For some instances, these methods were able to also determine the direction of the transfer, which was evaluated against expert knowledge.

3 Phylogenetic Inference in Computational Biology

Phylogenetic methods and sequence analysis techniques from computational biology offer a rich toolbox for studying the history of languages from a mathematical point of view. In order to successfully apply these methods, it is necessary to give a short introduction to some basics.

In section 3.1, Markov Models are introduced as they lay the groundwork for several techniques later on. Section 3.2 follows up with an explanation of alignment and algorithmic alignment techniques. While the first part of section 3.2 introduces basic alignment techniques, the second part establishes the concept of Pair Hidden Markov models. Pair Hidden Markov models connect the idea of alignment and Markov models. Section 3.2.3 presents the statistical alignment framework. This framework takes an evolutionary point of view on the alignment problem. The section on models of

DNA evolution (section 3.3) introduces the idea of explicit mathematical models to study the evolutionary history of nucleotides. These models are used in a wide range of studies concerned with phylogenetic problems. Another important matter in the study of phylogenetics is the concept of a tree. The second but last section shortly defines this concept. The last section tries to exemplify how to use these models described so far for the issue of phylogenetic inference. Moreover, it introduces an algorithm commonly used in this framework and the important Pulley Principle.

3.1 Background on Markov Models

Markov models provide a powerful and important machinery to several disciplines. Markov chains and, in particular, Hidden Markov models offer a convenient mathematical structure for the analysis of sequential data. Especially in the area of Speech Recognition, Hidden Markov models are frequently applied. As stated in the famous introduction to Hidden Markov models by Rabiner, as early as the late 1960s and early 1970s, these models are used in computational approaches to language processing (Rabiner, 1989). But also for computational biology, Hidden Markov models play an important role. The sequential nature of, for example, genetic data lends itself naturally to these kinds of models (cf. Durbin et al., 2001, for an overview of some applications in computational biology).

3.1.1 Markov Chains

Markov chains implement a particular kind of model generating sequences.[1] Sequential data is present in a lot of scientific areas. The DNA sequence in biology is sequential and words or texts in linguistics are sequential, just to name two. Quite generally the

[1] This introduction to Markov Chains is based on Durbin et al. (2001)

different elements within a particular sequence appear with a certain probability. Thus, the probability of a sequence S of length n could be written as follows.

$$\begin{aligned} P(S) &= P(x_n, x_{n-1}, \dots, x_1) \\ &= P(x_n|x_{n-1}\dots x_1)P(x_{n-1}|x_{n-2}\dots x_1)\dots P(x_1) \end{aligned} \tag{3.1}$$

The second line of equation 3.1 can be derived via several applications of the chain rule of probability. The way Markov chains model sequences is such that the probability of the n-th segment just depends on the segment $n-1$. This is called the Markov property. Applying the Markov property to the example above considerably simplifies the formula.

$$P(S) = P(x_n|x_{n-1})P(x_{n-1}|x_{n-2})\dots P(x_2|x_1)P(x_1) \tag{3.2}$$

To explain this idea, consider the following example. Suppose there is a DNA sequence. The letters which are the elements of this sequence are `A`, `C`, `G` and `T`. Each element of the sequence is one of these four elements. Figure 3.1 on page 17 shows a graphical representation of the described model. The black arrows indicate which symbol can follow which. In this model, all transitions between the states representing `A`, `C`, `G` and `T` are possible. Each arrow is associated with a probability parameter indicating how probable it is to move from one state to another. An important point here is that all the transition probabilities of leaving a state, independent of their destination, sum up to one. In the representation of the likelihood of a particular sequence which, according to the Markov model can be factorized as in equation 3.2, the probability $P(x_n|x_{n-1})$ is exactly the one present at the arrows in the model. The same equation includes the probability of the first symbol $P(x_1)$. Instead of modeling these probabilities explicitly, a *begin* state can be added. Each Markov chain will start at the *begin* state and the probability to observe the first symbol is the probability to move from the *begin* state to

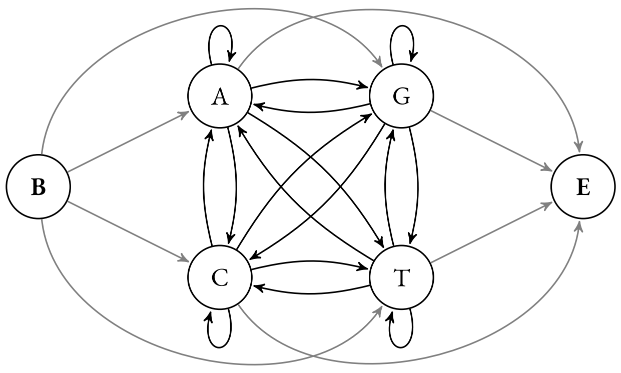


Figure 3.1: Example of a Markov chain which can generate a DNA sequence. The begin (**B**) and end (**E**) state are added and transition out and into the respective states are colored in gray (cf. Durbin et al., 2001, p. 49).

the state of the respective symbol. Similarly an *end* state can be added. A move into this state will terminate the Markov chain. Both the *end* and the *begin* state are not modeled via proper symbols in the sequence but are rather treated as silent states.

3.1.2 Hidden Markov Models

A lot of work on Hidden Markov models was done in the area of speech recognition. Originally developed by Leonard E. Baum and his colleagues (Baum and Petrie, 1966; Baum and Eagon, 1967; Baum and Sell, 1968; Baum et al., 1970; Baum, 1972), a famous introduction to Hidden Markov models is due to Rabiner (1989).

Hidden Markov models extend the idea of Markov chains. In a Markov chain, the state the model currently is in is observable. In a Hidden Markov model (HMM) this is not the case. To put it slightly differently, in a Markov Chain the sequence of symbols equals the symbols of states. Therefore, for a Markov chain, there is one stochastic process, namely the one generating the sequence of states. In an HMM, there is a second stochastic process at work. In addition to the process responsible for a sequence of states, there is another stochastic process governing the production of the observable sequence of symbols (Rabiner, 1989).

A formal definition of an HMM requires five parts (Rabiner, 1989, p. 260):

1. The number of states N in the model, with $S = \{s_1, s_2 \dots s_N\}$ as the set of states
2. The number of distinct observations M per state, with $W = \{w_1, w_2 \dots w_M\}$ as the set of all symbols
3. The set of state transition probabilities A with $a_{ij} \in A$ being the probability to move from state s_i to state s_j
4. The observation symbol probability distribution B in state s_j, where $b_j(v)$ is the probability of observing symbol v at state s_j
5. The initial state distribution π

Although the sequence of states underlying the stochastic process is hidden, there are often some clues available to which and how many states there should be. The distinct observations per state are symbols we observe, so M is the discrete alphabet size for that state. The state transition probabilities indicate the probabilities of moving from one state to the next. If for all $s_i, s_j \in S$ the transition probability $a_{ij} > 0$ then each state can be reached from every other state in a single step. As for Markov chains, all the transition probabilities of leaving a state should sum up to one, i.e., $\sum_j a_{ij} = 1$. The probability distribution of symbol observations indicates how likely a given state produces a symbol. Lastly, the initial state distribution π indicates how likely it is to start in any of the states in S. If there is a designated start state s_S, then $\pi(s_S) = 1$, if there is no such designated state, then $\Sigma_{s_i}^{S} \pi(s_i) = 1$ (cf. section 3.1.1). Given appropriate values for N, M, A, B and π an HMM can be used to generate a sequence of observations $O = o_1 o_2 o_3 \dots o_l$, where each o_i is in V and l is the length of the sequence O. A closer observation of the five points above shows that the number of states N and the size of the alphabet M are already implicitly defined through A and B. I adapt the more compact definition by Rabiner (1989)

$$\lambda = (A, B, \pi) \tag{3.3}$$

to define an HMM. HMMs are accompanied by a range of well-studied algorithms which help to answer three basic questions these models can be asked (cf. Rabiner, 1989).

1. What is the probability of the observed sequence given the model?
2. Which sequence of states ς maximizes the probability of the observed sequence?
3. Which configuration of model parameters accounts best for a given set of observations?

There are three basic algorithms which can provide an answer to these three questions. By using probability theory, the three problems can be formulated as follows.

$$P(O|\lambda) \tag{3.4}$$

$$\underset{\varsigma}{\operatorname{argmax}}\, P(O, \varsigma|\lambda) \tag{3.5}$$

$$\max P(O|\lambda). \tag{3.6}$$

The first problem aims at finding the model which best matches an observed sequence. This can be viewed as a problem of deciding between different models, given a particular sequence of observations. The second problem slightly differs from the first in that it is not aimed at selecting the model with the best fit, but rather at uncovering the hidden state path which is responsible for creating sequence O. The last problem is best viewed from the perspective of training. Given a particular model topology, the question is which parameters maximize the probability of the sequence being generated by that model. Hence, the task is to find a set of transition and emission probabilities which are optimal under certain criteria. This third problem is particularly interesting for most applications of HMMs since it is aimed at creating the best model for a given set of data (cf. Rabiner, 1989).

The first problem comes down to collecting the probabilities of all state paths of length l and their respective probability to emit sequence O. The probability of a particular state sequence can easily be calculated by considering the state transition probabilities and the probability that this sequence of states emits O can be obtained via the probability distributions of symbols per state. The probability of the observation sequence given the model is then calculated by summing over all the state paths. A simple enumeration of the sequences is computationally too heavy since the number of state sequences "increases exponentially with the length of the sequence" (Durbin et al., 2001, p. 57). However, the forward algorithm can be used to solve this problem. The forward algorithm is based on a dynamic programming paradigm. The idea is to calculate the probability based on a recursive condition. Suppose the probability of the sequence O up to symbol o_i ending in state $s_n \in S$ given the model is known ($f_n(i)$), then the probability to end at any given state $s_k \in S$ with observation o_{i+1} can be calculated from this quantity. This can be formulated as a recursive equation.

$$f_n(i) = b_n(o_i) \sum_k^N f_k(i-1) a_{kn} \tag{3.7}$$

(cf. Durbin et al., 2001)

Notice, that in this equation it is not $i+1$ computed from i, but rather i from $i-1$. This equation underlies the forward algorithm. The final probability $P(O|\lambda)$ is then the sum over the product of the final forward variables ($f_k(o_l)$) and their transition probabilities to the end state.

The backward algorithm works in a similar fashion to the forward algorithm and can also be used to solve the first problem. Instead of considering the probability of the partial sequence from o_1 to o_i, the backward algorithm considers the probability of the partial sequence from o_i to o_l. A recursive equation, similar to 3.7, can be formulated for the backwards probability β:

$$\beta_n(i) = \sum_{k}^{N} a_{nk} b_k(o_{i+1}) + \beta_k(i+1) \tag{3.8}$$

(cf. Durbin et al., 2001)

The algorithm which can be used to solve the second problem is very much related to the forward algorithm. The way the problem is phrased above, the optimal sequence of states is the one with the highest probability. Suppose the sequence of states with the highest probability up to observation o_i which ends in state s_n, $v_n(i)$, is known, then the value for o_{i+1} can be obtained in a similar fashion to the forward algorithm by replacing the summation with the maximization operation.

$$v_n(i+1) = b_n(o_{i+1}) \max_k (v_k(i) a_{kn}) \tag{3.9}$$

(cf. Durbin et al., 2001)

This recursive equation will find the state sequence with the highest probability. However, to obtain the best sequence of states in the end, it is necessary to store pointers to the state maximal previous state k. Through backtracking these pointers, the optimal sequence of states can be reconstructed. The algorithm which implements this is called the Viterbi algorithm. It is, similar to the forward and the backward algorithm, a dynamic programming algorithm, which can calculate the optimal state sequence in an easy fashion.

The answer to the last problem mentioned by Rabiner (1989) can be solved via several algorithmic techniques. As soon as the path through the state space is unknown, as it is usually the case, there is no analytical equation which can be used. The algorithms then rely on some form of iterative optimization procedure. In principle, there is a wide array of tools available. However, in the realm of HMMs the Baum-Welch algorithm (Baum, 1972), which belongs to the area of Expectation-Maximization algorithms (Dempster et al., 1977), is typically used. The idea of this approach is pretty straightforward. Firstly, given the current model parameters consider all probable paths for the training sequences. Secondly, derive new model parameters given the paths just calculated. This procedure is repeated until a stopping criterion is reached. Using this approach, it can be shown that the overall log likelihood of the model will increase. However, there is no guarantee that the maximum which is found is the global maximum.

To derive a more formal description of the Baum-Welch algorithm, suppose that $\mathcal{A}_{kn}$ is the number of transitions from state s_k to state s_n in the training data and that $E_k(x)$ is the number of emissions of symbol x from state s_k in the training data, where $x \in W$. The posterior probability that in the state path ς the state at position i was n ($\varsigma_i = n$) and at position $i + n$ was k ($\varsigma_{i+1} = k$) given the sequence O and the current model parameters can be written as follows.

$$P(\varsigma_i = n, \varsigma_{i+1} = k|O, \lambda) = \frac{f_n(i)a_{nk}b_n(o_{i+1})\beta_k(i+1)}{P(O|\lambda)} \tag{3.10}$$

(cf. Durbin et al., 2001)

The expected number of transitions from n to k, $\mathcal{A}_{nk}$, can be derived from this equation by summing over all positions in the sequence and all sequences in the training set.

$$\mathcal{A}_{kn} = \sum_j \frac{1}{P(O^j|\lambda)} \sum_i f_n^j(i) a_{nk} b_n(o_{i+1}^j) \beta_k^j(i+1) \tag{3.11}$$

(cf. Durbin et al., 2001)

Similarly, the expected number of times a symbol o is emitted from state k can be calculated.

$$E_k(x) = \sum_j \frac{1}{P(O^j|\lambda)} \sum_{\{i|o_i^j = x\}} f_k^j(i) \beta_k^j(i) \tag{3.12}$$

(cf. Durbin et al., 2001)

For both equations 3.11 and 3.12, f_n^j denote the forwards variable in state n for sequence j; this also holds for β_n^j. The maximum likelihood estimators for the actual transition from state kn, a_{kn}, and emission of x from state k, $e_k(x)$ can now be calculated.

$$a_{kn} = \frac{\mathcal{A}_{kn}}{\sum_l \mathcal{A}_{kl}} \tag{3.13}$$

(cf. Durbin et al., 2001)

$$e_k(x) = \frac{E_k(x)}{\sum_y E_k(y)} \tag{3.14}$$

(cf. Durbin et al., 2001)

Using these relations, a new set of model parameters can be estimated. The Baum-Welch algorithm does this iteratively. Starting with random model parameters, the forward and backward tables are computed, using the respective algorithm. These tables can be used to estimate new A and E parameters. Again, the forward and backward tables are calculated. This process is repeated until a convergence criterion is reached. Typical convergence criteria are a sufficiently small change in the parameter values or a sufficiently small change in the overall log likelihood.

3.2 Background on Alignment

Throughout the following, capital letters $A, B, C, \ldots$ will denote sequences constructed over an alphabet Σ. The elements of the sequences are indexed such that a_1 is the first element of sequence A and a_n is the last element of A which is of length n. The elements of the sequences are ordered such that a_i directly precedes a_{i+1}.

Sequence comparison has a long-standing tradition in several scientific areas, such as linguistics, computer science or bioinformatics, among others. The overarching question is if two sequences are related via a common cause. This question is mostly answered on the basis of an alignment. Following Kruskal (1983) and List (2014b), an alignment is a matrix in which sequences are arranged such that proper matches are listed in the same column.[2]

Definition 3.1. Let A and B be sequences over Σ. Let $-$ denote a *space*. An alignment of A and B is a matrix with two rows, such that

1. the entries in the matrix consist of $\Sigma \cup \{-\}$
2. the first row contains the symbols of A in order and the second row the symbols of B, respectively
3. one or more spaces, $-$, can appear between consecutive symbols of A or B,
4. each column contains at least one letter of Σ.

(cf. Chao and Zhang, 2008)

[2] List (2014b) provides a good introduction to the issue of sequence comparison and general concepts of alignment in a linguistic setting.

n	m	N
2	1	5
4	2	41
8	4	3649
10	10	8097453
106	106	$< 10^{80}$
107	107	$> 10^{80}$

Table 3.1: Number of alignments N for two sequences of length n and m. Where 10^{80} supposedly is the number of protons in our universe (cf. Torres et al., 2003; McPherson, 2008).

Deriving such an alignment is far from trivial. Given two sequences of length n and m respectively, the number of possible alignments N grows pretty fast. Table 3.1 illustrates this behavior. Torres et al. (2003) derive a formula to calculate the number of possible alignments.

$$N = \sum_{k=0}^{min\{n,m\}} = 2^k \binom{m}{k} \cdot \binom{n}{k} \tag{3.15}$$

N grows rapidly with increasing m and n. Typical sequence lengths in biological applications are usually greater than 200. Thus, finding the optimal alignment by an enumeration method is not feasible. This problem can be solved by scoring alignments based on an additive score. Assuming such a scoring scheme, the dynamic programming paradigm can be used for finding the optimal alignment.

In historical linguistics, sequence alignment takes a lot of its inspiration from computational biology. In this area there are two approaches which need to be discussed; score based and statistical alignment. Score based methods try to maximize a similarity score or minimize a distance between two or more sequences, based on an abstract set of parameters. Statistical alignment approaches aim to maximize or minimize said scores on the basis of an underlying model of evolution or phylogeny (Lunter et al., 2005). The aim of score based alignment is to best describe the data, while statistical alignment strives to find an explanatory model. Several of the basic techniques and terms which are necessary for statistical alignment are best introduced using score based alignment.

Additionally, score based alignment techniques have been very successful in computational historical linguistics (see section 4.1).

3.2.1 Score Based Alignment

Score based alignment techniques come in two different but similar forms, distance or similarity based (Smith et al., 1981). Both methods aim to generate an alignment which either minimizes the distance or maximizes the similarity. Since both approaches are so similar, I will concentrate on the similarity based approaches to explain the concept.

Finding the optimal alignment requires the definition of a metric on alignments. Generally, there are four cases which can occur:

1. Two identical symbols appear in the same column (first column in (1)),
2. two non-identical symbols appear in the same column (third column in (1)),
3. a symbol in the first row is paired with a space or gap sign in the second row (second column in (1)),
4. a symbol in the second row is paired with a space or gap sign in the first row (last column in (1)).

The first case, namely an identical pair of symbols in a column, is called a *match*; the second case, namely a different pair of symbols in a column, is called a *substitution*. The third and fourth case are also termed deletion and insertion respectively. The alignment in (1) exhibits three matches, one substitution and two insertions, and one deletion.

```
(1)   A G T A – C –
      A – C A G C C
```

Quite generally, an alignment can be represented in terms of an *alignment graph*. The vertices of the alignment graph for two sequences A and B of length m and n can be thought of as the lattice points (i, j) where $0 \leq i \leq n$ and $0 \leq j \leq m$. The

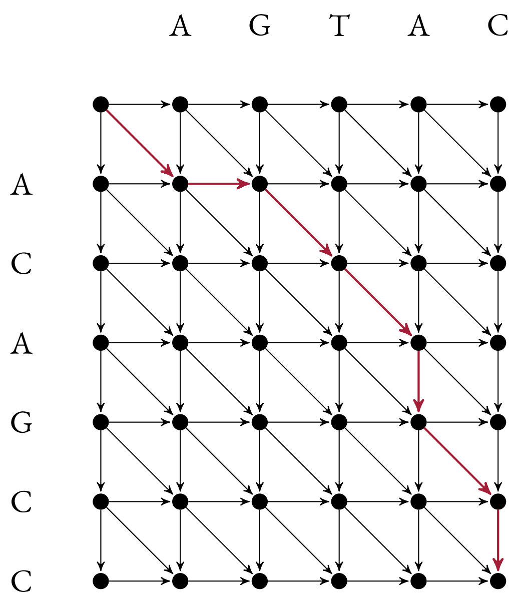


Figure 3.2: Example of an alignment graph. The bold red lines represent the alignment shown in (1).

edges of this graph are directed such that there is an edge between (i, j) and (i', j') if $0 \leq i' - i \leq 1$ and $0 \leq j' - j \leq 1$. Thinking in terms of a lattice, the edges can either be diagonal ($i' - i = 1$ and $j' - j = 1$), horizontal ($i' - i = 1$ and $j' - j = 0$) or vertical ($i' - i = 0$ and $j' - j = 1$). A diagonal arrow indicates a substitution or a match, a vertical arrow a gap in the first and a horizontal arrow a gap in the second sequence. Each path through this grid which starts at the top left cell and ends at the bottom right cell corresponds to a particular alignment (Chao and Zhang, 2008). Figure 3.2 shows an example of such an alignment graph. Each alignment between the two sequences is represented by a particular path through this graph.

The *global alignment* problem is to find the alignment or path through this grid, which maximizes the similarity score. A score can only be maximized according to a given scheme. Conceptually, matches should have the highest score, substitutions a medium score and gaps in either string should be penalized. Needleman and Wunsch (1970) proposed an algorithm which guarantees to find the alignment with the highest similarity score given a particular scoring scheme. Their method is based on a dynamic programming paradigm. The scoring scheme Needleman and Wunsch proposed in their paper assigns a positive weight, 1, to matches, a neutral weight to substitutions, 0, and a slight penalty, -1, to deletions and insertions. The optimal alignment which has the highest similarity score is the one which maximizes the sum of all alignment moves. Although the algorithm by Needleman and Wunsch does not directly operate on a graph, the concept of the alignment graph can be transferred to their approach.

For two sequences A and B of length n and m, respectively, let $f(i, j)$ denote the alignment for the sequences A and B up to a_i and b_j, respectively. There are three possible moves which can happen next according to the alignment graph: a diagonal, a horizontal or a vertical move. The horizontal or vertical move corresponds to the introduction of a gap. Such a move is penalized by -1. The diagonal move can either be a substitution or a match, so scored with a 0 or a 1, respectively. Having enumerated these possibilities, the following recurrence can be stated.

$$f(i,j) = \max \begin{cases} f(i-1,j) - 1 \\ f(i,j-1) - 1 \\ f(i-1,j-1) + 1 & \text{if } a_i = b_j \\ f(i-1,j-1) & \text{if } a_i \neq b_j \end{cases} \tag{3.16}$$

The corner case is $f(0, 0)$ which is set to 0. The cases $f(0, j)$ and $f(i, 0)$ successively introduce gaps with scores $-1 \cdot j$ and $-1 \cdot i$, respectively. An immediate improvement of this method is a more advanced scoring scheme. Such a scheme ideally introduces a more fine-grained analysis into the scoring of matches and substitutions via a scoring function. A scoring function $\sigma(i, j)$ assigns an individual score to the pair i, j based on some external characteristics. Introducing sigma and a variable β for a gap move simplifies the recurrence.

$$f(i,j) = \max \begin{cases} f(i-1,j) - \beta \\ f(i,j-1) - \beta \\ f(i-1,j-1) + \sigma(a_i, b_j) \end{cases} \tag{3.17}$$

A famous extension to this algorithm is due to Gotoh (1982). Instead of a penalizing constant β for introducing gaps, gaps are weighted by their length. Thus, β will be split into a non-negative gap opening penalty α and a non-negative gap extension penalty $\gamma \cdot i$, with i being the length of the gap. This way of parameterizing gap penalties is known as *affine gap penalties*. The idea of separating constant gap penalties from affine gap penalties is that opening a new gap should be more expensive than extending an existing one. The practical implementation of the method proposed by Gotoh (1982) is still done in the dynamic programming paradigm but requires a three-dimensional matrix instead of a two-dimensional matrix. The three-dimensional matrix is of size $(n+1) \times (m+1) \times 3$ where n and m are the length of the two sequences and three resembles the possible alignment moves of substitution/match, insertions, and deletions. These three dimensions are labeled S, I and D. The algorithm is then formulated as follows (cf. Chao and Zhang, 2008; Gotoh, 1982):

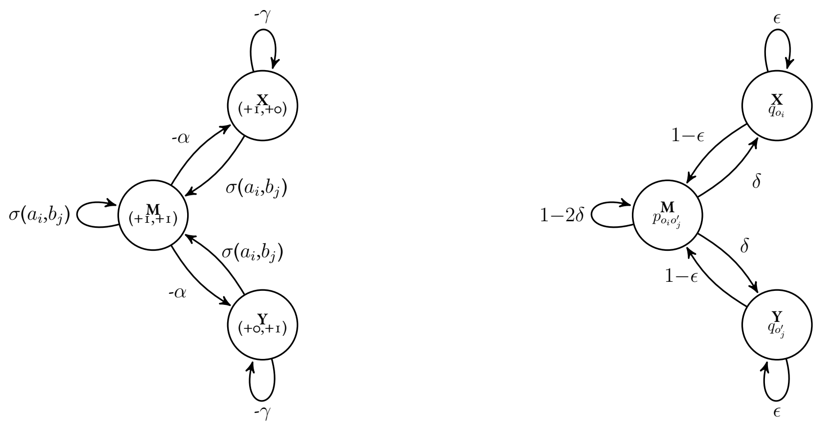
$$\begin{aligned} D(i,j) &= \max \begin{cases} D(i-1,j) - \gamma \\ S(i-1,j) - \alpha - \gamma \end{cases} \\ I(i,j) &= \max \begin{cases} I(i,j-1) - \gamma \\ S(i,j-1) - \alpha - \gamma \end{cases} \\ S(i,j) &= \max \begin{cases} D(i,j) \\ I(i,j) \\ S(i-1,j-1) + \sigma(a_i, b_j) \end{cases} \end{aligned} \tag{3.18}$$

In the calculation of $D(i,j)$ and $I(i,j)$, the difference between gap opening and gap extension penalties are visible and so is the difference to the algorithm published by Needleman and Wunsch (1970). The extension of a gap is penalized with γ and the opening is penalized by α and γ. The initialization of $S(0,0) = 0$ is straightforward and unproblematic.[3] The dimensions I and D are initialized as $I_{0,k} = -\gamma_k - \alpha$ for $1 \leq k \leq m$ and $D_{k,0} = -\gamma_k - \alpha$ for $1 \leq k \leq n$. Equation 3.19 shows another formulation of the algorithm which is used repeatedly.

$$
\begin{aligned}
D(i,j) &= \max \begin{cases} D(i-1,j) - \gamma \\ S(i-1,j) - \alpha - \gamma \end{cases} \\
I(i,j) &= \max \begin{cases} I(i,j-1) - \gamma \\ S(i,j-1) - \alpha - \gamma \end{cases} \\
S(i,j) &= \max \begin{cases} D(i-1,j-1) + \sigma(a_i,b_j) \\ I(i-1,j-1) + \sigma(a_i,b_j) \\ S(i-1,j-1) + \sigma(a_i,b_j) \end{cases}
\end{aligned}
\tag{3.19}
$$

In terms of an actual alignment, this reformulation ends an insertion or deletion event with an explicit match rather than doing it silently. This version of the algorithm, in a slight extension of the term, will be called *Needleman-Wunsch algorithm* throughout the remainder.

[3]As Flouri et al. (2015) note in the original paper of Gotoh, there is a slight mistake regarding the initialization of the dynamic programming table. I adapt the corrected version according to Flouri et al. (2015).



(a) Finite State automaton representation of the dynamic programming algorithm for pairwise alignment

(b) Probabilistic version of the FSA for affine gap alignment.

Figure 3.3: FSA and probabilistic model of affine gap alignment (cf. Durbin et al., 2001, p. 81).

3.2.2 Pair Hidden Markov Models

The *Needleman-Wunsch algorithm* is based on a dynamic programming paradigm.[4] It is possible to construct a *finite state automaton* (FSA) which corresponds to the dynamic programming algorithm for pairwise alignment (Karp and Held, 1967; Durbin et al., 2001). For the algorithm for affine gap alignment described in the previous section (cf. equation 3.19), the corresponding FSA is shown in figure 3.3a. The FSA is to be understood such that the transitions between the states carry out a score increment and the states specify the increment value of the index in the respective sequence. As before, for each state, there is a variable at position (i, j). The respective values are calculated as the maximum of the incoming transitions. The FSA can be converted into a Hidden Markov model with some straightforward changes. The first change is concerned with the score assigned to matches and gapped symbols. If the states in the HMM are understood as either emitting a pair of symbols (state M) or emitting a symbol in one string and a gap in the other (states X and Y), the scores turn into emission probabilities. For the HMM formulation of the alignment approach, there needs to be

[4]The argumentation in this part follows Durbin et al. (2001)

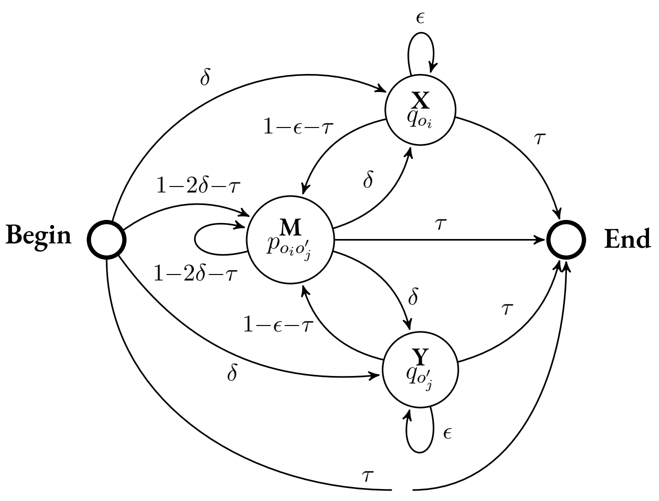


Figure 3.4: Pair HMM (cf. Durbin et al., 2001, p. 82), where δ, ϵ and τ are some transition probabilities and $p_{o_i o'_j}$, q_{o_i} and $q_{o'_j}$ are distributions over pairwise and single symbol emissions.

a probability distribution over the emission of two symbols and distributions for emitting a single symbol and a gap in either string. For reasons of symmetry, the distribution assigned to the X and Y state are identical. Thus, there are two distributions for symbol emissions, one for a pair of symbols and one for a symbol and a gap.

The second modification deals with the transition between the states. It is necessary that the probabilities of leaving a state sum to one. Again for reasons of symmetry, there are just two free parameters. These transformations will result in a probabilistic version of the FSA (see figure 3.3b). However, the probabilistic version of the FSA does not represent a full HMM. For a proper HMM, an explicit *begin* and *end* state are necessary. In comparison to the probabilistic FSA, an additional parameter is required to account for the transitions out of the begin and into the end state. Figure 3.4 shows the resulting model.

In comparison to the HMMs presented before, the emissions of this model happen in two output streams. Hence, there is a pair of output streams instead of just one. Taking this into account, this version of Hidden Markov models will be called a Pair Hidden Markov model (PHMM). As this model is a Markov model, the algorithms explained in section 3.1.1 can be transformed readily such that their structure matches the structure of PHMMs. After the incorporation of the necessary changes, the algorithms can be used to answer the questions posed by Rabiner (1989) in the realm of PHMMs (see page 19 of this dissertation).

The Viterbi algorithm can be used to find the most probable path through a model. In the case of PHMMs, this path corresponds to the most probable alignment. Equation 3.20 shows the recursion relations for the three states of the PHMM in terms of proper probabilities.

$$
\begin{aligned}
Y(i,j) &= q_{b_j} \max \begin{cases} Y(i,j-1)\epsilon \\ M(i,j-1)\delta \end{cases} \\
X(i,j) &= q_{a_i} \max \begin{cases} M(i-1,j)\delta \\ X(i-1,j)\epsilon \end{cases} \\
M(i,j) &= p_{a_i,b_j} \max \begin{cases} M(i-1,j-1)(1-2\delta-\tau) \\ X(i-1,j-1)(1-\epsilon-\tau) \\ Y(i-1,j-1)(1-\epsilon-\tau) \end{cases}
\end{aligned}
\tag{3.20}
$$

(cf. Durbin et al., 2001)

These equations already look strikingly similar to the recurrence relations of the Needleman-Wunsch algorithm. Durbin et al. (2001) show that via the use of log-odds alignment, the equivalent of the Needleman-Wunsch algorithm can be derived. These log-odds can be derived by using a proper random model. The probability of two sequences under the random model is the likelihood that the two sequences were generated randomly. While the probability that two sequences are related given the PHMM can be calculated via the recurrences shown above, the likelihood of two sequences under the random model R is shown in equation 3.21, the respective model is shown in figure 3.5,

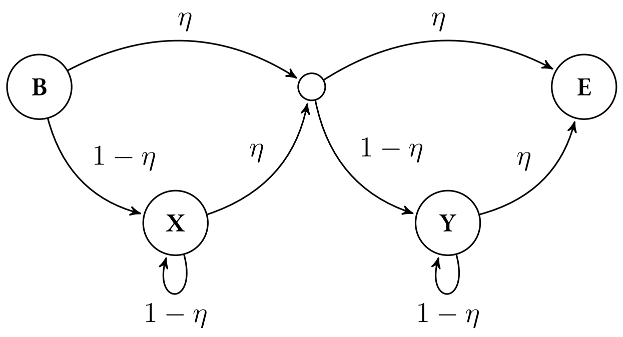


Figure 3.5: Random Model (cf. Durbin et al., 2001, p. 83); note the silent state which serves as a hub between the states X and Y, which ensures, that both sequences are generated independently.

where η is the state transition probability in the random model and q_{a_i} the equilibrium probability of symbol i in sequence a.

$$P(a, b|R) = \eta^2(1-\eta)^{n+m} \prod_{i=1}^{n} q_{a_i} \prod_{j=1}^{m} q_{b_j}. \tag{3.21}$$

In essence, this equation calculates the likelihood as the product of the probabilities of each symbol sequence and corrects these products by the factor $1-\eta$. It can be shown that the expected length of two sequences under the random model is $\frac{1-\eta}{\eta}$ (Borodovsky and Ekisheva, 2006). Thus, there is a correction for the length of the sequences incorporated in the random model. By using the results from the random model, the Viterbi algorithm can be adjusted to produce the optimal log-odds alignments (cf. Durbin et al., 2001).

After having stated the Viterbi algorithm, the forward and the backward algorithm for PHMMs are easily obtainable. The relationship between these algorithms carries over from the HMM case. As for HMMs, the forward or backward algorithm can be used to calculate the probability of the observed pair of sequences given the model. Since a path through the PHMM represents a particular alignment of two sequences, the result of the forward algorithm returns the probability that the two sequences are related given any alignment. Instead of finding the path that maximizes the probability, the probability of the two sequences given the model is calculated by summing over all paths through the model (see 3.22).

$$\begin{aligned}
Y(i,j) &= q_{b_j}\left[Y(i,j-1)\epsilon + M(i,j-1)\delta\right] \\
X(i,j) &= q_{a_i}\left[X(i-1,j)\epsilon + M(i-1,j)\delta\right] \\
M(i,j) &= p_{a_i,b_j}[M(i-1,j-1)(1-2\delta-\tau)+ \\
&\qquad X(i-1,j-1)(1-\epsilon-\tau)+ \\
&\qquad Y(i-1,j-1)(1-\epsilon-\tau)]
\end{aligned} \tag{3.22}$$

(cf. Durbin et al., 2001)

Correspondingly, the backward algorithm can as well be formulated for PHMMs (cf. Mackay, 2004; Wieling, 2007). Wieling (2007) also shows how the Baum-Welch algorithm, which has the forward and backward algorithm as its parts, can be adapted for PHMMs.

Criticism of Score Based Alignment

Following the argumentation of Thorne et al. (1992), score based alignment is subject to two sources of subjectivity, which may influence the performance of this method.

The first point of criticism is the scoring function used to score matches or substitutions. It is far from obvious how much better or worse, in terms of an absolute score, a particular substitution is compared to another. However, there are attempts to circumvent this specific source of subjectivity. On the one hand, there are empirical approaches in computational biology such as the BLOSUM (Henikoff and Henikoff, 1992) and PAM matrices (Dayhoff et al., 1978). However, these matrices were developed on the basis of already existing alignments. Thus, these matrices do not readily solve this problem. Due to the large number of data and some manual correction of the alignments prior to building these matrices, the problem is mitigated. Another approach to removing subjectivity from the creation of the scoring function is based on an algorithmic estimation of these parameters. In an early instance, Fitch and Smith (1983) use a Monte Carlo method to estimate adequate model parameters. In other areas, algorithmic approaches mostly based on the Expectation-Maximization paradigm (Dempster et al., 1977) were developed to estimate the parameters of the scoring function (cf. Ristad and Yianilos, 1998; Wieling et al., 2009; Jäger, 2013, see also chapter 4). Although these approaches estimate the parameters algorithmically and thus remove subjectivity from the equation, these approaches are merely descriptive. In the essence of the approach of Thorne et al. (1992), a model which has an underlying (evolutionary) interpretation is required. The approaches listed above try to describe the data rather than explaining it.

The second point of subjectivity Thorne et al. identified is more subtle. It is rather connected to the algorithmic aspect than to the scoring of substitutions or matches. Thorne et al. (1992) note that "it is not clear that the form of the gap penalty should be $G_i = a + bi$" (Thorne et al., 1992, p. 3). Without stating the evolutionary assumptions underlying the process which can be modeled using this formula, such a function is arbitrary. The reason for choosing this formulation is rather on the algorithmic side than supported by any theoretical considerations (cf. Thorne et al., 1992). The approach Thorne et al. took was to develop a statistical framework which makes explicit assumptions about the evolutionary process. This proposal was then developed into the statistical alignment framework (see section 3.2.3).

3.2.3 Statistical Alignment

The statistical alignment approach is based on two papers by Thorne et al. (termed TKF91 and TKF92, respectively) (Thorne et al., 1991, 1992). As the name of this approach suggests, alignment is handled in a statistical framework which allows parameter inference, hypothesis testing, and analysis of uncertainty (Lunter et al., 2005). In

$\bullet$ G $\star$ A $\star$ T $\star$ T $\star$ A $\star$ C $\star$ A $\star$

Figure 3.6: Sequence modeled as consisting of links ($\star$) and nucleotides.

comparison to the score-based alignment approaches, statistical alignment starts with explicit assumptions about the way DNA sequences evolve. Starting from there, the parameters used for the actual alignment are then a function of the divergence time and the speed or rate of evolution. This fits well with the view, that "a sequence alignment is designed to exhibit the evolutionary correspondence between different sequences" (Thorne et al., 1991, p. 114). However, the statistical alignment approach has two components. One component describes the insertion-deletion process which sequences undergo. Thus, it is necessary to develop a model describing how sequences evolve with respect to their length. The second component is an explicit evolutionary model of the substitution process. This aspect of statistical alignment is concerned with an explicit evolutionary model of how bases evolve over time. There are several models belonging to the second component. They will be discussed in further detail in section 3.3. The remainder of this section rather focuses on the aspect of the insertion-deletion process and only mentions the substitution process in abstract terms if necessary.

Thorne et al. (1991) propose an evolutionary model of sequence insertion and deletion events. The model is designed as a continuous time Markov process of insertion and deletion events. These events originate from so-called *links*. There is a link at the start and the end of a sequence, as well as a link between each nucleotide (see figure 3.6). The leftmost link is designated as an *immortal link* ($\bullet$). All the other links $\star$ are normal links and occur by definition to the right of a base. For a sequence of length n there are then $n + 1$ links, one to the right of each base and the immortal link. The actual process of insertion and deletion is formulated in terms of these links instead of the bases. The fate of each link is independent of the fates of the other links. Each link is subject to a birth-death process where all links can spawn another link with probability λ and all normal links ($\star$) can die with probability μ. Whenever a link dies, the base on its left will die with it and whenever a new link spawns, it will be accompanied with a new base to its left. By definition, a newborn link can never be an immortal link and will always be a normal one. According to Thorne et al. (1991), the probability of more than one insertion or deletion event happening in the same instant is so small that it can be ignored. Thus, for a sequence of length n there can be $n + 1$ links which can give birth to a new link with rate λ. A sequence of length n will therefore grow at rate $(n + 1)\lambda$. On the other hand, there are only n links which can die at rate μ, since the immortal link by definition cannot die. A sequence of length n then decreases to length $n - 1$ with the rate of $n\mu$. The immortal link being immune to deletion prevents sequences from growing to an infinite length or being of length 0. If additionally the death rate is larger than the birth rate, the equilibrium distribution of the sequence length γ_n is the geometric distribution (Thorne et al., 1991).

$$\gamma_n = \left(1 - \frac{\lambda}{\mu}\right)\left(\frac{\lambda}{\mu}\right)^n \tag{3.23}$$

This gives an explicit model of the insertion-deletion process. Together with a process of base substitutions, the likelihood that one sequence evolves into another can be calculated.

To illustrate the structural aspect of statistical alignment, assume two sequences A and B which have length n and m respectively. Consider the example from (1) (repeated here as (2)).

(2)
```
A G T A – C –
A – C A G C C
```

As mentioned above, the focus of this section is on the insertion deletion process and not on the evolutionary model of the substitution process. Since the TKF process is also formulated in terms of links, the same alignment will just be represented in terms of these links.

(3) ● ⋆ ⋆ ⋆ ⋆ – ⋆ –
● ⋆ – ⋆ ⋆ ⋆ ⋆ ⋆

The task is to derive a likelihood of how a sequence of links of length $m + 1$ evolves from a sequence of links of length $n+1$.[5] The particular alignment shown in (3) could have been generated by a multitude of evolutionary histories. The goal of statistical alignment is to consider all these histories in the calculation of the likelihood of this alignment. Assume that the sequence in the top row is the ancestral sequence and the sequence in the second row is the descendant. In order to derive the likelihood of this particular alignment structure, the fate of each of the ancestral links needs to be considered. The final likelihood is then the joint probability of all these fates. For each link, there are in principle two fates over time and a third for the immortal link:

1. There are l descendant links from a normal one, and the original link is one of them,
2. there are l descendant links from a normal one, and the original link dies,
3. the immortal link has l descendants including itself.

The likelihoods of the different scenarios are $p_l^H(t)$, $p_l^N(t)$ and $p_l^I(t)$ respectively, with t as the evolutionary time and l as the number of descendant links. For the given example, the immortal link gives rise to zero new nucleotides. The fate of the first normal link is described by the first scenario, i.e., $p_l^H(t)$. This can be done for each of the links in the ancestral sequence. The full likelihood $P(\alpha|\theta)$ with α as the alignment structure and θ as the model parameters is then:

$$P(\alpha|\theta) = p_0^I(t)p_1^H(t)p_0^N(t)p_1^H(t)p_2^H(t)p_2^H(t). \tag{3.24}$$

Thorne et al. (1991) or Lunter et al. (2005) give differential equations for the probabilities of the different fates. As already noted above, this particular alignment can

[5]The 1 is added because of the immortal link present in every sequence.

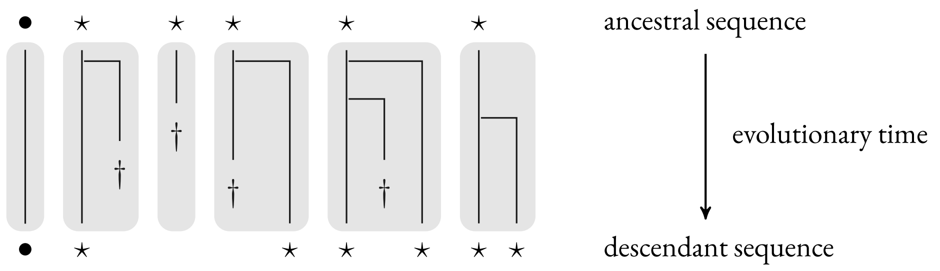
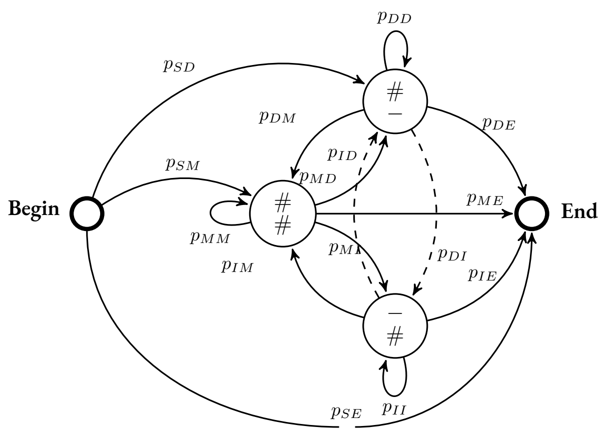


Figure 3.7: Exemplary representation of an evolutionary history which is compatible with the alignment shown in (3). The † represents the death of the particular residue (cf. Lunter et al., 2005).

be the result of several different evolutionary histories. One such scenario is shown in figure 3.7. This is just one history which produces the alignment displayed in (3). Statistical alignment considers all possible histories in the formulation of the probabilities. Thus, it remains neutral regarding the particular evolutionary history and rather quantifies how probable it is that any evolutionary history generated a particular alignment. Instead of using these differential equations, the process described by Thorne et al. (1991) can also be formulated in terms of a Hidden Markov model (cf. Lunter et al., 2005; Holmes and Bruno, 2001; Hein, 2001; Metzler et al., 2001). The formulation of the TKF91 model in terms of a Markov model makes the model in itself more comprehensible and the idea of the model becomes a bit clearer (see figure 3.8 on page 41).

$$
\begin{aligned}
B_\tau &= \lambda\beta(\tau) \\
E_\tau &= \mu\beta(\tau) \\
N_\tau &= (1 - e^{-\mu\tau} - \mu\beta(\tau))(1 - \lambda\beta(\tau)) \\
H_\tau &= e^{-\mu\tau}(1 - \lambda\beta(\tau)) \\
I_\tau &= 1 - \lambda\beta(\tau) \\
\beta(t) &= \frac{1 - e^{(\lambda-\mu)t}}{\mu - \lambda e^{(\lambda-\mu)t}}
\end{aligned}
\tag{3.25}
$$

(cf. Lunter et al., 2005)



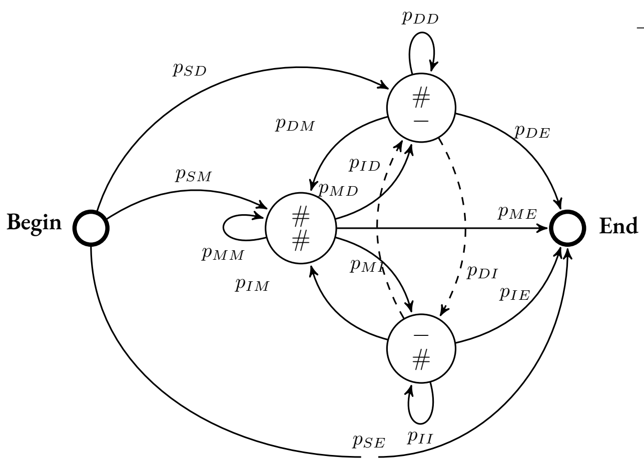
arc	transition probability
p_{MM}	$B_\infty H_\tau$
p_{MI}	B_τ
p_{MD}	$B_\infty(N_\tau + E_\tau(1 - B_\tau))$
p_{ME}	$(1 - B_\tau)(1 - B_\infty)$
p_{II}	B_τ
p_{IM}	$B_\infty H_\tau$
p_{ID}	$B_\infty(N_\tau + E_\tau(1 - B_\tau))$
p_{IE}	$(1 - B_\tau)(1 - B_\infty)$
p_{DD}	$E_\tau B_\infty$
p_{DM}	$\frac{E_\tau H_\tau B_\infty}{(N_\tau + E_\tau(1-B_\tau))}$
p_{DI}	$\frac{N_\tau}{(N_\tau + E_\tau(1-B_\tau))}$
p_{DE}	$\frac{E_\tau(1-B_\tau)(1-B_\infty)}{(N_\tau + E_\tau(1-B_\tau))}$
p_{SM}	$\frac{B_\infty H_\tau}{1-B_\tau}$
p_{SD}	$B_\infty\left(\frac{N_\tau}{1-B_\tau} + E_\tau\right)$
p_{SE}	$1 - B_\infty$

Figure 3.8: Hidden Markov model formulation of the TKF91 insertion deletion process. The transition parameters are as proposed by Lunter et al. (2005). The evolutionary time separating the two sequences is τ and ∞ represents the equilibrium state, thus an infinite amount of time.

The actual probabilities on the arcs in the Markov model formulation follow from the differential equations from Thorne et al. (1991) and are also shown in equation 3.25.

It is important to note that the model developed by Thorne et al. (1991) is defined such that per unit in time, there is just one base which can be inserted or deleted. The model cannot handle the insertion or deletion of longer snippets. Thorne et al. further developed their model such that the insertion or deletion of longer snippets are possible. It turns out that the model described above emerges as a special case from the extended model.

The TKF92 model extends the TKF91 model such that the deletion or insertion process does not just delete or insert single links or bases but rather entire fragments. The insertion and deletion process on the links is the same in both models. Therefore, the extension of the model is done just in terms of integrating the idea of fragments. To properly formulate the model of insertions and deletions, it is necessary to define the distribution which governs the size of the fragments. Thus, let $h(n)$ be the probability



arc	transition probability
p_{MM}	$r + (1-r)B_\infty H_\tau$
p_{MI}	$(1-r)B_\tau$
p_{MD}	$(1-r)B_\infty(N_\tau + E_\tau(1-B_\tau))$
p_{ME}	$(1-r)(1-B_\tau)(1-B_\infty)$
p_{II}	$r(1-r)B_\tau$
p_{IM}	$(1-r)B_\infty H_\tau$
p_{ID}	$(1-r)B_\infty(N_\tau + E_\tau(1-B_\tau))$
p_{IE}	$(1-r)(1-B_\tau)(1-B_\infty)$
p_{DD}	$r(1-r)E_\tau B_\infty$
p_{DM}	$(1-r)\frac{E_\tau H_\tau B_\infty}{(N_\tau+E_\tau(1-B_\tau))}$
p_{DI}	$(1-r)\frac{N_\tau}{(N_\tau+E_\tau(1-B_\tau))}$
p_{DE}	$(1-r)\frac{E_\tau(1-B_\tau)(1-B_\infty)}{(N_\tau+E_\tau(1-B_\tau))}$
p_{SM}	$\frac{B_\infty H_\tau}{1-B_\tau}$
p_{SD}	$B_\infty\left(\frac{N_\tau}{1-B_\tau} + E_\tau\right)$
p_{SE}	$1 - B_\infty$

Figure 3.9: Hidden Markov model formulation of the TKF92 insertion-deletion process. The transition parameters are as proposed by Lunter et al. (2005). The evolutionary time separating the two sequences is τ and ∞ represents the equilibrium state, so the time parameter tends to infinity. The definition of the parameters in the table is the same as in equation 3.25.

that a link comes with $n \geq 1$ bases. Thorne et al. (1992) postulate that $h(n)$ comes in the form of a geometric distribution.

$$h(n) = (1-r)r^{n-1}$$
$$0 \leq r < 1, n \geq 1 \tag{3.26}$$

For $r = 0$ the TKF91 model emerges as a special case of the TKF92 model. A consequence of introducing the idea of fragments is that the distribution of sequence length does no longer follow the geometric distribution of the TKF91 model (cf. 3.23).

$$\gamma_n = \left(1 - \frac{\lambda}{\mu}\right)\frac{\lambda}{\mu}(1-r)\left[\frac{\lambda}{\mu}(1-r) + r\right]^{n-1} \tag{3.27}$$

In parallel to the TKF91 model, the parameters λ and μ are the birth and death rate. The TKF92 model can be formulated in terms of a Hidden Markov model as well (cf. figure 3.9). The difference between the Hidden Markov model formulation of the TKF91 and TKF92 process is rather straightforward. The transition parameters for the TKF92 model are derived from the TKF91 model by including the parameter governing the distribution of the fragment length r (cf. equation 3.26). Each transition probability is multiplied by $(1-r)$ and r is added to each self-transition (cf. Lunter et al., 2005). Only the transitions out of the begin state remain unaltered. Reformulating the TKF91 and TKF92 model in terms of Hidden Markov models opens up the toolbox of standard Markov Model algorithms, such as the forward or Viterbi algorithm.

While the TKF92 model is probably closer to reality than the TKF91 model, there are still two strong assumptions underlying the model which are potentially problematic. The first assumption is that insertion and deletion events cannot overlap and the second assumption is that the length of the insertions or deletions is geometrically distributed (Lunter et al., 2005; Thorne et al., 1992). A more general model, which relaxes in particular the first assumption, was developed by Miklós et al. (2004). This model, called 'long indel' model, allows overlapping insertion and deletion events. This relaxation may be desirable from a conceptual point of view but comes at a high computational price. Relaxing the non-overlap assumption results in an entanglement of neighboring nucleotides over time (cf. Lunter et al., 2005). Thus, the final probability cannot be factorized similar to the standard TKF91 or TKF92 model. Additionally, for the general form of the 'long indel' model "no closed form solution of the outcome probabilities are known, even for geometric indel length distributions" (Lunter et al., 2005, p. 381). There are still algorithms which can calculate an approximation to the 'long indel' model, although they are computationally more demanding than the TKF91 and TKF91 models.

Phylogenetic Inference under the Statistical Alignment Model

In conjunction with a proper model of base substitution, phylogenetic inference is possible. For certain specific trees, there are even algorithms allowing the direct calculation of the likelihood without resorting to sampling approaches. Steel and Hein (2001) present an algorithm which can calculate the likelihood of three sequences related by a star-shaped tree directly. Miklós (2002) improves the algorithm of Steel and Hein but also states, that it is only reasonable for about three sequences. For a small number of sequences, Hein (2001) presents an algorithm which can calculate the likelihood of a set of sequences evolving along the branches of a binary tree. In each of these cases, the authors note that the application of sampling-based methods, such as Markov Chain Monte Carlo methods, should be considered for real-sized problems.

However, there are at least two implementations available which model the use of sampling-based methods in conjunction with statistical alignment. The StatAlign software package (Novák et al., 2008) and the Bali-phy software package (Suchard and Redelings, 2006) implement the statistical alignment approach in conjunction with phylogenetic inference (Hein et al., 2003).

3.3 Models of DNA Evolution

As already noted previously, statistical alignment approaches use explicit models of DNA evolution to model similarities between DNA nucleotides. These models are not only used in the area of statistical alignment but also for already aligned sequences of DNA data. Since the interpretation of these alignments is that the nucleotides appearing in the same column have a common origin, models of DNA evolution can be used to model how these nucleotides evolve over time. These models can then be used for the task of phylogenetic reconstruction of trees based on a set of functionally equivalent and aligned DNA sequences. Naturally, the state space of these models consists of the four bases. Figure 3.10 (page 45) illustrates the state space of such a model. The arrows indicate possible changes of the bases.

Actually modeling these transitions between the states has lead to a wide number of different models. Figure 3.11 on page 45 gives a graphical overview of several of these models. As the graph already suggests, the Jukes-Cantor model (JC in the graph) (Jukes

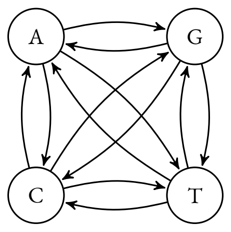
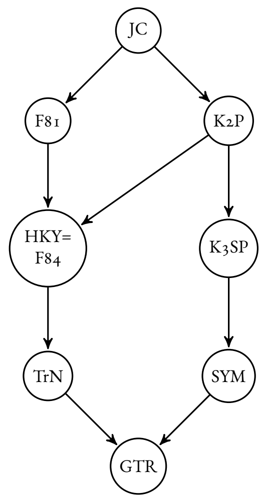


Figure 3.10: Symbolic representation of the evolutionary process underlying the evolution nucleic acids.



JC	Jukes and Cantor (1969)
F81	Felsenstein (1981)
K2P	Kimura (1980)
K3SP	Kimura (1981)
HKY	Hasegawa et al. (1985)
F84	Felsenstein and Churchill (1996)
TrN	Tamura and Nei (1993)
SYM	Zharkikh (1994)
GTR	Lanave et al. (1984)

Figure 3.11: Overview of the relations between different models of DNA evolution. The arrows should be interpreted such that the model at the tip of the arrow is an extension compared to the model at the origin of the arrow (cf. Huson et al., 2010).

and Cantor, 1969) is the most basic one. One by one the different models relax the very restrictive assumptions of this model. The most general of these models is the General Time Reversible model (GTR in the graph) (Lanave et al., 1984). To give an introduction to this model, the model by Jukes and Cantor as well as the general time reversible model will shortly be reviewed. For a more in-depth introduction to these models see Felsenstein (2004).

3.3.1 The Jukes-Cantor Model

The Jukes-Cantor model is the earliest and most basic model of DNA evolution.[6] It also places the most restrictions on the parameters of the model. The model is defined such that each base has the same chance to change into any other base. In terms of the graph in figure 3.10, the weight of each arrow is identical. Consequently, the four bases are expected to appear with the same frequency. Assume the rate of substitutions between all bases is u thus, since all changes have the same chance, each arrow is associated with the weight $\frac{u}{3}$. The graph in figure 3.10 with the associated weights can be represented in terms of a matrix as in 3.28.

$$\begin{pmatrix} \cdot & \frac{u}{3} & \frac{u}{3} & \frac{u}{3} \\ \frac{u}{3} & \cdot & \frac{u}{3} & \frac{u}{3} \\ \frac{u}{3} & \frac{u}{3} & \cdot & \frac{u}{3} \\ \frac{u}{3} & \frac{u}{3} & \frac{u}{3} & \cdot \end{pmatrix} \tag{3.28}$$

Given a pair of aligned sequences, the distance between the two can be estimated by calculating the transition probabilities from one state to another, i.e., from one base to the aligned other. To ease the calculation, assume that there is an additional possible change from a base to itself with rate $\frac{u}{3}$. This would fill the diagonal of the matrix 3.28 with $\frac{u}{3}$. Instead of choosing from three bases in the case of a transition, the original base is a candidate as well, which results in four candidates. The rate of change is then not u anymore but rather $\frac{4}{3}u$. This will be the same process as before since the probability to change to a different base remains $\frac{u}{3}$. The evolutionary time separating the two sequence is t. So the expected number of transition events is $\frac{4}{3}ut$. Since the time t is taken to be continuous, the whole process can be modeled via a Poisson distribution. Thus, in each tiny segment of time dt the probability of an event is $\frac{4}{3}u\ dt$. The probability

[6]The description of the Jukes-Cantor model loosely follows Felsenstein (2004).

of no event happening is then the zero term of the Poisson distribution (cf. Felsenstein, 2004).

$$e^{-\frac{4}{3}ut} \tag{3.29}$$

Hence, the probability of at least one event happening is the complement of this quantity.

$$1 - e^{-\frac{4}{3}ut} \tag{3.30}$$

Since a consequence of the model's parametrization is that all bases have the same frequency, the probability of a base turning into another base is defined as

$$\frac{1}{4}(1 - e^{-\frac{4}{3}ut}). \tag{3.31}$$

Hence, the probability of seeing any other base after evolutionary time t is then three times the quantity defined in 3.31.

$$\frac{3}{4}(1 - e^{-\frac{4}{3}ut}) \tag{3.32}$$

For a given alignment this equation can be used to derive a maximum likelihood estimate of the evolutionary time t separating the two sequences.

3.3.2 The General Time Reversible Model

The general time reversible (GTR) model (Lanave et al., 1984) allows the maximal amount of flexibility while still being mathematically tractable.[7] Similar to the Jukes-Cantor model, the General Time Reversible model is *reversible*. Reversibility describes the property that the probability of starting with x at the beginning of a branch and ending at y after time t is the same as starting with y and having x after the same amount

[7]The model by Rodríguez et al. (1990) is in theory even more flexible in the choice of parameters. However, Felsenstein (2004) and Rodríguez et al. (1990) point out their needs to be more work done to make this model useful.

	A	G	C	T
A	$\cdot$	$\pi_G\alpha$	$\pi_C\beta$	$\pi_T\gamma$
G	$\pi_A\alpha$	$\cdot$	$\pi_C\delta$	$\pi_T\epsilon$
C	$\pi_A\beta$	$\pi_G\delta$	$\cdot$	$\pi_T\eta$
T	$\pi_A\gamma$	$\pi_G\epsilon$	$\pi_C\eta$	$\cdot$

Table 3.2: Parametrization of the general time reversible model. The table should be read such that the base in the row changes to the base in the column so, e.g., $\pi_C\delta$ is the instantaneous rate of change from G to C. This formulation of the GTR model is due to Lanave et al. (1984) (cf. Felsenstein, 2004).

of time. Suppose π_x and π_y are the equilibrium frequencies of x and y and $P(x|y,t)$ is the probability of starting with y and ending with x after time t, and vice versa for $P(y|x,t)$. Formally speaking, reversibility is then defined as in 3.33.

$$\pi_x P(y|x,t) = \pi_y P(x|y,t) \tag{3.33}$$

Note that reversibility does not assume that $P(x|y,t) = P(y|x,t)$. Although reversibility is mathematically convenient, there is no reason to believe that the actual underlying biological process is reversible. Besides this drawback, it appears that reversible models match the actual observed data really closer. Therefore, the true underlying biological process may actually be close to reversible (cf. Felsenstein, 2004). The formulation of the actual rates of change in the general time reversible model is shown in 3.2. The table shows the rates of change from one base to another according to the GTR model. Compared to the Jukes-Cantor model, this model has six different substitution parameters and unequal base frequencies. The Jukes-Cantor model emerges as a special case from the GTR model if $\pi_A = \pi_G = \pi_C = \pi_T$ and $\alpha = \beta = \gamma = \delta = \epsilon = \eta$. To calculate actual transition probabilities from the rates of change, a rate matrix Q is necessary. The rate matrix for the GTR model is shown in equation 3.34, where φ, χ, ψ and ω are each the sum of the other entries in their row.

$$Q = \begin{pmatrix} -\varphi & \pi_G\alpha & \pi_C\beta & \pi_T\gamma \\ \pi_A\alpha & -\chi & \pi_C\delta & \pi_T\epsilon \\ \pi_A\beta & \pi_G\delta & -\psi & \pi_T\eta \\ \pi_A\gamma & \pi_G\epsilon & \pi_C\eta & -\omega \end{pmatrix} \tag{3.34}$$

The values in the matrix should be normalized by the sum of the off-diagonal values multiplied by the probability to start with the respective base such that for a unit of time, there is one change. The matrix of transition probabilities after time t $P(t)$ can be calculated by exponentiating the matrix A.

$$P(t) = e^{At} \tag{3.35}$$

The entries in the resulting matrix are the probabilities to observe the base in the row at the beginning of the branch of the tree and the base in the column at the end (the interpretation of rows and columns carry over from table 3.2). The estimation of the values for the frequencies and rates can be done via a numerical optimization technique. There are mainly two software suits available, which among other things, also estimate the optimal branch lengths and tree structures. There is BEAST on the one hand (Suchard et al., 2018) and MrBayes (Huelsenbeck and Ronquist, 2001; Ronquist and Huelsenbeck, 2003) on the other hand.

3.4 Trees

An important concept for phylogenetic inference in computational biology and historical linguistics is the concept of a *tree*. The tree as a mathematical concept is a unifying object for disciplines concerned with the study of evolutionary processes. Trees can be used to answer questions of relatedness and or time scales. Evolutionary time, as mentioned several times above already, can easily be modeled by the means of a tree. The evolutionary time separating two taxa can be defined as the length of the branches connecting the two (a more formal definition is given below). Therefore, the following section will shortly introduce this concept. The following overview is based on Steel (2016).

Formally speaking, trees are special instances of graphs.

Definition 3.2. An undirected graph $G = (V, E)$ consists of set of vertices V and a set of edges $E \subseteq \binom{V}{2}$.

While, in theory, there is no restriction on the size of V and E only finite sets of V and E, are considered in the following. Two vertices or nodes n and n' in a graph are adjacent if there is a set $e = \{n, n'\}$ in E. A node n and an edge e are incident if $n \in e \in E$. Since the set of edges is defined as having sets of single elements as their sole members, it follows that the edges in this graph are undirected. There are two more consequences following this definition of graphs: fist loops are excluded as well as parallel edges. Both of these points follow from defining the set of edges as a set of sets with a cardinality of two. The degree of a node is the number of edges which are incident with this node. A path in a graph is defined as a sequence of different nodes $n_1 \dots n_k$ such that each node is adjacent to the next in the sequence. The length of a path is defined as the number of edges on the path. If there is a path with $k \geq 3$ such that n_1 and n_k are adjacent, the path constitutes a cycle of length k. A tree is then defined as follows.

Definition 3.3. A tree $T = (V, E)$ is a graph that is connected and has no cycles.

In comparison to an undirected graph, there is a unique path between any two vertices u and v in a tree T. Nodes which have a degree of 1, so just one incident edge, are called *leaves*. Vertices, which are not leaves are called *interior*. The definitions of graphs and trees so far (definition 3.2 and definition 3.3, respectively), do not employ any notion of direction on their edges. Ergo in terms of evolution and time, for two connected nodes, there is no way to tell which one is the ancestor and which the descendant. To get a formal grasp on this, it is necessary to include the notion of directionality. In order to add directionality, the definition of edges as a set of sets needs to be changed.

Definition 3.4. Directed graphs consist of a set of vertices V and a set of *directed* edges $A \subseteq (V \times V)$. A directed edge (u, v) starts at u and ends at v.

The set of edges for a directed graph is no longer a set of sets but rather a set of ordered pairs, where the order of the elements determines the direction of the edge. Instead of the degree of an edge, the *in-degree* and *out-degree* are distinguished. The in-degree of a node n is the number of edges ending at n, i.e., the number of pairs where n is the second element, and the out-degree of n is the number of edges starting at n. A *directed acyclic graph* (DAG) is further on defined as a graph without a directed path from a vertex to itself. If a node n can be reached via a directed path from node u in a DAG, then n is called a *proper descendant* of u. There are two fundamental properties of DAGs by which they are characterized: 1. If a directed graph G is acyclic, then it has at least one vertex of out-degree zero and at least one vertex of in-degree zero; 2. G is acyclic if and only if there exists a total ordering on the vertices of G that satisfies the following condition: if v_0 is a proper descendant of v, then v is strictly less then v_0 in that ordering. Following up on DAGs, a rooted tree will be defined as follows.

Definition 3.5. A rooted tree is a tree $T = (V, E)$, where a vertex $v \in V$ is distinguished as a root vertex and all edges are directed away from this vertex. The root vertex is the only node with in-degree 0. In addition, for a rooted binary tree, each non-leaf vertex has out-degree 2.

Rooted trees are an important concept for the study of evolutionary processes. With rooted trees, the notion of the latest common ancestor of two elements can be defined.

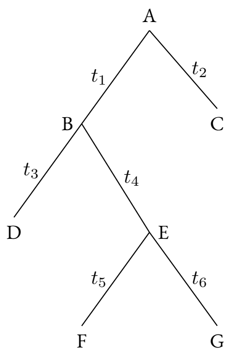


Figure 3.12: Example for a rooted tree with branch lengths

A node l is the latest common ancestor of the nodes u and v if u and v are proper descendants of l and there is no other node l' which is a proper descendant of l and of which u and v are also proper descendants. Also, the notion of evolutionary time can be defined properly. To quantify the evolutionary time, it is necessary to include the notion of branch length. The branch length can be defined via a function b which assigns a real-valued length to each $e \in E$. Therefore, the evolutionary time separating two nodes u and v is the sum of all branch lengths on the path connecting u and v.

3.5 Felsenstein's Algorithm and the Pulley Principle

Having stated the mathematical theory behind trees and the models of DNA evolution in previous sections, it is possible to calculate the likelihood of a tree. A computationally efficient way to calculate this likelihood is provided by Felsenstein's algorithm. Felsenstein developed this algorithm in two papers (Felsenstein, 1973, 1981). The algorithm is based on the idea to factorize the likelihood of a complex tree into the likelihoods of smaller sub-trees. In the same instance, Felsenstein pointed out that the reversibility property of the models of DNA evolution allowed a useful property of the estimation of evolutionary trees to be established.

As a general idea, the likelihood of a tree should be the product of the probability of changes along the branches of the tree. Let s_X denote the state of node X, i.e., the nucleotide present at this node, and π_X be the prior probability of s_X. For the tree in figure 3.12 on page 52, the likelihood would be computed as the probability of change along the branches times the prior probability of the state of the root nodes (cf. Felsenstein, 1981).

$$\begin{aligned} L = {} & \pi_A P(s_B|s_A,t_1)P(s_C|s_A,t_2)P(s_D|s_B,t_3) \\ & P(s_E|s_B,t_4)P(s_F|s_E,t_5)P(s_G|s_E,t_6) \\ & \text{(cf. Felsenstein, 1981)} \end{aligned} \tag{3.36}$$

Since the state of the root and all the other internal nodes is usually unknown, the different states of the bases need to be summed over.

$$\begin{aligned} L = \sum_{s_A}\sum_{s_B}\sum_{s_C}\sum_{s_E} & \pi_A P(s_B|s_A,t_1)P(s_C|s_A,t_2)P(s_D|s_B,t_3) \\ & P(s_E|s_B,t_4)P(s_F|s_E,t_5)P(s_G|s_E,t_6) \\ & \text{(cf. Felsenstein, 1981)} \end{aligned} \tag{3.37}$$

Felsenstein mentions that for four possible states at the internal nodes, the resulting expression for a tree wit n leaves would consist of 2^{n-2} terms. It turns out that the full likelihood can be computed much more efficiently. The summation signs can be moved 'rightwards' such that the scope of the different summation signs mirrors the topology of the tree. The new structure of the formula indicates that the likelihood of the tree can be evaluated by going from the leaves to the root of the tree. This particular way of passing through the tree is also known as post-order traversal. This allows the definition of a recursive mechanism to calculate the likelihood of a tree. Assume that the node n has two proper descendant nodes l and r, whose incident branches have length t_l and

t_r respectively, and that x and y are possible states of a node. Then the mechanism can be stated generally by the following formula.

$$L_n(s) = \left(\sum_x (P(x|s,t_l)L_l(x))\right)\left(\sum_y (P(y|s,t_r)L_r(y))\right) \tag{3.38}$$

(cf. Felsenstein, 1981)

Where $L_n(s)$ is the probability of observing everything on the descendants of n given that n is in state s. At the leaves of the tree, $L_z(s)$ is 1 for the nucleotide which is actually present and 0 for the other states. The likelihood of the entire tree can then be calculated as follows.

$$L = \sum_{s_\rho} \pi_{s_\rho} L_r(s_\rho) \tag{3.39}$$

(cf. Felsenstein, 1981)

For all possible states s_ρ at the root ρ the probability of all possible scenarios given s_ρ is computed. The full likelihood is then the sum over all these probabilities times the prior probability of s_ρ, π_{s_ρ}. The development of the method focused on just one particular site. The likelihood for a nucleotide sequence is obtained by summing over the likelihood of each site, where the likelihood of a site is computed by the means of equation 3.39.

The second important mechanism which is due to Felsenstein is the *Pulley Principle*. This principle is a consequence of the reversibility property of the models of DNA evolution. He shows that the likelihood term in equation 3.38 is only affected by the sum of t_l and t_r. As long as $t_l + t_r$ remains unaltered, $L_n(s)$ is not affected by a changing of the branch length. Thus, even though the algorithm works on the basis of a rooted tree, what is actually estimated is an unrooted tree.

4 Automatic Cognate Identification and Phylogenetic Inference

The detection of cognate word pairs is an important aspect of computational historical linguistics.[1] Recent years have seen a surge in the number of publications in the field of computational historical linguistics due to the availability of word lists for a large number of languages of the world as well as cognate databases for Austronesian or Indo-European (Brown et al., 2013; Bouckaert et al., 2012; Greenhill and Gray, 2009).

The availability of word lists (without cognate judgments) has sparked different lines of research. On the one hand, scholars like Rama and Borin (2015) and Jäger (2015) experimented with different weighted string similarity measures for the purpose of inferring the family trees of the languages of the world, without explicit cognate identification. Their research can be counted towards the area of distance based methods for

[1] The first part of the chapter is based on Rama et al. (2017).

	ALL	AND	ANIMAL	...
English	ol	End	Enim3l	...
German	al3	unt	tia	...
French	tu	e	animal	...
Spanish	to8o	i	animal	...
Swedish	ala	ok	y3r	...

Table 4.1: Example of a word list for five languages belonging to Germanic (English, German and Swedish) and Romance (Spanish and French) subfamilies transcribed in ASJP alphabet.

phylogenetic inference. On the other hand, List (2012a) proposed a cognate clustering system that combines handcrafted weighted string similarity measures and permutation tests for the purpose of automated cognate identification. In a different approach, Hauer and Kondrak (2011) experimented with linear classifiers like SVMs (Support Vector Machines) for the purpose of identifying cognate clusters. Jäger et al. (2017) also used SVMs for the task of cognate identification. Finally, Rama (2015) uses string kernel inspired features for training a SVM linear classifier for pair-wise cognate identification.

In another line of research, the inference of phylogenetic trees using Bayesian methods (Yang and Rannala, 1997) in computational historical linguistics uses data which is annotated for cognacy by experts. This research explores the idea of reconstructing language trees (e.g., Gray and Atkinson, 2003; Bouckaert et al., 2012) or the inspection of sound change events (Bouchard-Côté et al., 2013). The phylogenetic inference methods which operate on the character level require cognate judgments which are only available for a small number of well-studied language families such as Indo-European and Austronesian. For instance, the ASJP database provides Swadesh word lists transcribed in a uniform format for more than 60% of the languages of the world. An example of such a word list is given in table 4.1. To further facilitate the use of phylogenetic methods, reliable cognate judgments for a lot of language families are necessary. While the supervised methods of cognate identification (e.g., Hauer and Kondrak, 2011; Jäger et al., 2017; Rama, 2015) require some annotated data for training, well performing unsupervised methods are desirable to deal with the large amount of un-annotated data. Thus, one task at hand is to automatically cluster words that show genealogical relationship.

As noted by Hauer and Kondrak (2011) and tested by Rama et al. (2018), availability of a reliable multilingual cognate identification system can be used to supply the cognate judgments as an input to the phylogenetic inference algorithms. Such an approach connects these two steps of the comparative method which have been treated as separate previously in computational historical linguistics.

In section 4.1, there is a short review of some relevant literature in cognate detection and alignment as well as phylogenetic inference. Section 4.3 presents an unsupervised system for the detection of cognate classes from un-annotated data. The system is compared against other systems from the literature and its performance is critically assessed. Section 4.4 presents the paper by Rama et al. (2018) in some detail. Their experiments use automatically inferred cognate sets for the purpose of estimating language trees. The experiments by Rama et al. (2018) are partly based on the cognate classes which were detected with the methods presented in section 4.3.

4.1 Sequence Alignment and Cognate Detection in the Literature

Sequence alignment methods and cognate detection approaches have made their way into the literature in several ways already. The two areas most relevant to the current study are dialectometry and computational historical linguistics. I discussed some of the approaches above.

It has to be noted at this point that the notion of a *cognate* in the area of computational historical linguistics often employs a slightly narrower definition of the word than in traditional historical linguistics (cf. section 2.1 of this thesis). The definition of cognacy in computational historical linguistics is limited to words describing the same concept. The reason for this restriction is first and foremost rooted in the design of the experiments as well as the lack of *cross-concept* cognacy annotated datasets. What this means in practice is that a cognate pair like English *bone* and German *Bein* (leg) will be completely ignored although they are cognate in the traditional sense. Throughout the remainder, cognacy is used in its more narrow, computational sense.

PHMMs are used in computational dialectology (Wieling et al., 2007) and early approaches to computational historical linguistics (Mackay and Kondrak, 2005). Both studies use PHMMs to compute word similarity. While the early study by Mackay and Kondrak had the same goal as the study presented here, i.e., to identify cognate word pairs, Wieling et al. used PHMMs to incorporate weighted sequence distance measures into the comparison of Dutch dialects.

The Needleman-Wunsch algorithm has also drawn the attention of several scholars in dialectometry and computational historical linguistics. As discussed above, the actual scoring scheme is a point of focus. Deriving a scoring scheme based on the concept of pointwise mutual information (PMI) (Church and Hanks, 1990) is fairly popular (see section 4.3.1 for details).Wieling et al. (2009, 2012) employed the idea of a PMI based scoring scheme. While Wieling et al. (2009) compared among others the results of PMI based alignment and PHMMs, Wieling et al. (2012) used PMI scores to measure acoustic differences. In computational historical linguistics, Jäger (2013) used the PMI based alignment method for computing the string similarity using the ASJP database.

Kondrak (2000) introduced a dynamic programming algorithm for computing the similarity between two sequences. The scoring scheme is based on articulatory phonetic features determined by Ladefoged and Johnson (2014). The structural part of the algorithm is similar to the Needleman-Wunsch algorithm, but instead of considering only the symbol at the previous position, the ALINE algorithm also considers the symbols at previous position (see also section 4.3.5). Kondrak introduced this extension, to account for cases where a symbol in one string corresponds to two symbols in the other string. The author evaluated his algorithm on a list of English-Latin cognates.

List (2012a) introduced a system known as LexStat (described in section 4.3.4) that is sensitive to segment similarities and chance similarities due to borrowing or semantic shift. The author tests this system on a number of small-sized datasets (consisting of less than 20 languages) for the purpose of cognate identification and reports that the system performs better than Levenshtein distance. The LexStat approach utilizes the idea of sound classes. The sound class alignment (SCA) (List, 2012c,b), which LexStat uses, is build on the idea proposed by Dolgopolsky (1964, 1986). The idea of sound classes is also present in the development of the scoring scheme used in the ALINE method. In a recent paper, List et al. (2016) explore the use of InfoMap (Rosvall and Bergstrom, 2008) for the detection of partial cognates in subgroups of Sino-Tibetan language family. The authors compare the performance of average linkage clustering against InfoMap and find that InfoMap performs better than average linkage clustering.

As mentioned above already, Hauer and Kondrak (2011) trained a linear SVM on word similarity features and use the SVM model to assign a similarity score to the word pair. For each meaning, a word pair distance matrix is computed and supplied to the average linkage clustering algorithm for inferring cognate clusters. The authors observe that the SVM trained system performs better than a baseline that judges the similarity of two words based on the identity of the first two consonants. Jäger et al. (2017) took a similar approach to Hauer and Kondrak (2011) in that they used a supervised SVM approach in conjunction with a clustering algorithm to infer cognate classes. In the study of Jäger et al., the authors used more elaborate word similarity measures than in the study of Hauer and Kondrak as well as another clustering algorithm. The studies listed above tested similar datasets using different experimental settings. For instance, Hauer and Kondrak (2011) trained and tested on a subset of language families that were provided by Wichmann and Holman (2013). At the same time, the LexStat system, although widely used, seems not to be evaluated on all the available language families.

Moreover, the PMI-LANG (Jäger, 2013), has not been evaluated at the task of unsupervised cognate clustering. Both systems, LexStat and PMI-LANG, were only used as part of a pipeline, which then was evaluated on cognate clustering (Jäger et al., 2017).

In a completely different approach, Bouchard-Côté et al. (2013) used their method which was designed to reconstruct word forms from ancient languages to infer cognate sets. In comparison to the other techniques described above, they rely on the presence of a family tree. This is not practical for a lot of other language families then the one studied by Bouchard-Côté et al. since the tree structure for many language families is not known beforehand.

4.2 Phylogenetic Inference in the Literature

The literature on phylogenetic inference using computational methods can be divided along the previously discussed line of character based and distance based methods. The character based methods work on the basis of discrete characters and use computational approaches which are based on explicit models of character evolution. These models are often versions of the DNA-evolution models from computational biology (see section 3.3). In a different fashion, the distance based methods calculate an actual distance between languages based on some features of the language, such as sound inventory sizes or aggregated alignment scores. The resulting distances are then supplied to a hierarchical clustering algorithm which outputs a phylogenetic tree.

The studies using character based methods for phylogenetic inference mainly evolve around the Indo-European and the Austronesian language family. The studies which are based on the Indo-European languages have set off a lively discussion about the age and origin of the Indo-European language. This discussion in the linguistic literature is very much intertwined with research in archeology and related disciplines. The two contenders are the Steppe-hypothesis (Anthony, 2010; Haak et al., 2015, among others) and the Anatolian hypothesis (Renfrew, 1987; Gray and Atkinson, 2003, among others). Both theories of origin come with different age estimates as well. While the Steppe-hypothesis predicts the birth date of the Indo-European languages to lay between 4500-3500 BCE, the Anatolian hypothesis predicts this language family to be about 3000 years older. Both theories have some properties which can be tested. The work by Gray and Atkinson (2003) and Bouckaert et al. (2012) present some evidence from linguistic dating and reconstruction of migration patterns in favor of the Anatolian hypothesis respectively. Using the same dataset and slightly different constraints, the studies by Chang et al. (2015) and Rama (2018) offered support for the Steppe-hypothesis, as their time estimates heavily favor an age estimate which is well within the range of the Steppe hypothesis. Despite their widely different results, these studies use the same tool box, to the extent that the algorithms which are used to estimate the parameters of their model are based on the Markov Chain Monte Carlo (MCMC) paradigm (see section 5.3 for an introduction).

The methods which Longobardi et al. (2013) and Ringe et al. (2002) used to study the phylogeny of Indo-European also use character data. However, the algorithms they use are different to the MCMC based algorithms used in the previously mentioned studies. The early study by Ringe et al. (2002) uses a mix of morphological, phonological and lexical characters as their data. The algorithm they used to infer a phylogenetic tree tries to minimize the incompatibility at internal nodes. Based on their dataset the algorithm is not able to produce a tree without conflict, but rather produces 18 different trees which the authors analyze further. Ringe et al. investigate these trees based on evidence from historical linguistics and explain the inconsistencies. Nevertheless, they conclude that they are able to recover major subgroups of Indo-European. Using different data and other methods, Longobardi et al. (2013) also investigated the Indo-European languages. They remained agnostic with regards to the origin of Indo-European and rather focused on exploring the usability of syntactic data to infer phylogenies. Based on syntactic features from the nominal phrase, such as the presence of a free Genitive, Longobardi et al. derive a distance measure which is then used as an input to a tree building algorithm. Thus, in terms of the character and distance based methods, their work can be counted among the distance based research. After inspecting the resulting trees, the authors claim that syntactic features provide a signal which is similar to the lexical signal used in other studies. Inconsistencies of their trees with gold standard trees are attributed to the implicational nature of some syntactic characters.

Using actual word forms, Jäger (2013, 2015) derives distances between languages using a weighted alignment technique (see section 4.3.1 for a description of the training method). The distance between languages is designed such that it measures the likelihood whether the degree of similarity between two languages could have come up by chance. These values are then used as an input to an algorithm which calculates the tree which assumes the minimum amount of evolution based on the input distance matrix. In comparison to the other methods described above, this approach does not need any data which is annotated by experts, it only needs semantically aligned and phonetically transcribed word lists. Additionally, this method is able to detect language families reliably.

4.3 Automatic and Unsupervised Cognate Identification

In this section I present a method aimed at automatically inferring cognate classes. The workflow will be to compute similarities between all the word pairs belonging to a meaning and then supplying the resulting distance matrix as an input to a clustering algorithm. The clustering algorithm groups the words into clusters by optimizing a similarity criterion. The similarity between a word pair can be computed using supervised approaches (cf. Hauer and Kondrak, 2011, among others) or by using sequence alignment algorithms such as Needleman-Wunsch (Needleman and Wunsch, 1970) or Pair Hidden Markov model approaches (Mackay and Kondrak, 2005).

This study uses an unsupervised approach to estimate word similarity. These similarity scores are then supplied to a clustering algorithm. The resulting clusters are evaluated against a gold standard to measure the performance. The focus is on the estimation of word similarities which are calculated by a version of the Needleman-Wunsch algorithm and a Pair Hidden Markov model (PHMM). The other two steps use standard methods to facilitate comparison against existing methods.

Afterwards, In introduce the models that are used to compute the sequence similarity, the training methods, and the clustering algorithms. The performance of the method is comprehensively evaluated with several experiments (secction 4.3.4). Finally, I present and discuss the results.

4.3.1 Models

While the idea of Pair Hidden Markov models and Needleman-Wunsch alignment was already laid out above, the following section discusses some changes to PHMMs and introduces an unsupervised mechanism to estimate a scoring scheme for Needleman-Wunsch alignment.

Both approaches, the Needleman-Wunsch algorithm and PHMMs, produce a similarity score, in the way they are set up in the current experiment. For the clustering algorithm, these similarity scores are converted into distances by using a sigmoid transformation:

$$f(x) = 1.0 - (1 + exp(-x))^{-1}. \tag{4.1}$$

This function maps all the similarity scores into the range of $[0, 1]$ (see figure 4.1 on page 64). Low similarity scores are mapped onto distance scores between 0.5 and 1, and high similarity scores are mapped onto distance scores between 0.5 and 0.

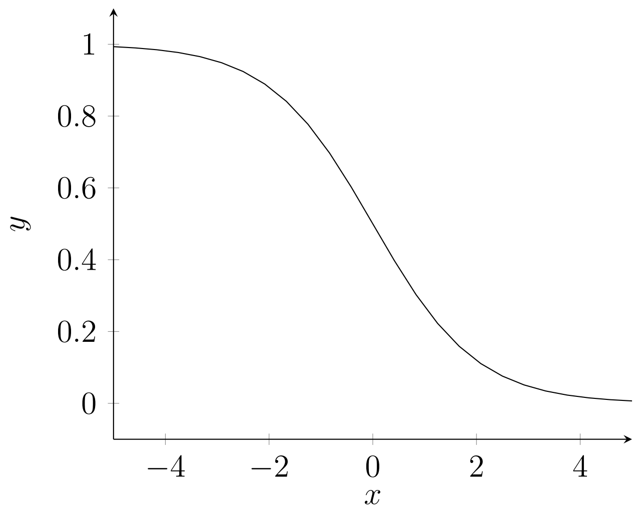


Figure 4.1: Example of a sigmoid function. In this particular case $y = 1.0 - (1 + exp(-x))^{-1}$.

PMI-Weighted Alignment

As discussed above, selecting the appropriate scoring scheme is an important aspect of using the Needleman-Wunsch algorithm. In comparison to computational biology, where there are pretty reliable and well crafted scoring schemes such as the BLOSUM (Henikoff and Henikoff, 1992) or PAM (Dayhoff et al., 1978) matrices, similar matrices are harder to obtain for linguistic data. There are exceptions such as the scoring scheme by Covington (1996) or Kondrak (2000). Jäger (2013) discusses the ALINE approach and compares it to PMI weighted alignment. He comes to the conclusion that PMI weighted alignment is superior to handcrafted schemes.

The approach taken here is to learn the scoring scheme from the data. Based on the idea that in cognate wordpairs sounds that have a close relationship should be aligned, it is possible to derive a scoring scheme. Mathematically, this is based on the idea of pointwise mutual information (Church and Hanks, 1990).[2] The PMI score for the two sounds i and j is defined as followed:

$$\text{PMI}(i, j) = \log \frac{p(i, j)}{q(i) \cdot q(j)} \tag{4.2}$$

where, $p(i, j)$ is the probability of i, j being matched in a pair of cognate words, whereas, $q(i)$ is the probability that an arbitrarily chosen segment in an arbitrarily chosen word equals i. A positive PMI value between i and j indicates that the probability of i being aligned with j in a pair of cognates is higher than what would be expected by chance. Conversely, a negative PMI value indicates that an alignment of i with j is more likely the result of chance than of shared inheritance. Jäger (2013) introduced this idea to computational historical linguistics. Slightly earlier and inspiring Jäger (2013), Wieling et al. (2009, 2012) used this measurement in dialectometry. Jäger (2013) brought up a method to infer the PMI score of two sounds from unaligned and unlabeled data. The training regime for the current study mirrors the one proposed by Jäger (2013). It can be described by the following steps.

1. Extract a set of word pairs that are probably cognate using a suitable heuristics. In this paper, we treat all word pairs belonging to the same meaning with a length normalized Levenshtein distance (LDN) below 0.5 as *probable cognates*.[3]

2. Align the list of probable cognates using the Needleman-Wunsch algorithm with the original settings of Needleman and Wunsch (1970) .

3. Extract aligned segment pairs and compute the PMI value for a segment pair using equation 4.2 and estimating probabilities as relative frequencies.

[2]In computational biology, the same measurement is called *log odds* score.
[3]See section 4.3.5 for an explanation of LDN.

4. Generate a new set of aligments using Needleman-Wunsch algorithm and the segment weights learned from step 3. For the gap penalties, we used the values proposed in Jäger (2013).

5. We iterate between step 3 and 4 until the average similarity between the two iterations does not change.

This procedure yields a scoring scheme, which is learned from the data and is not informed by prior expert knowledge. This scoring scheme can then be supplied to the Needleman-Wunsch algorithm and a PMI weighted similarity score for each wordpair can be computed.

Pair Hidden Markov Model

As previously discussed (see chapter 3.2.2), Pair Hidden Markov models (PHMMs) emerge from a probabilistic reformulation of the Needleman-Wunsch algorithm. Used in computational biology, PHMMs also found their way into linguistic applications (cf. Mackay and Kondrak, 2005; Wieling et al., 2007).

Mackay and Kondrak (2005) discussed the topology of PHMMs in their study. The topology of PHMMs in its original version shows no transition between the two gap states (states **X** and **Y** in figure 4.2 on page 67). The authors discuss this as a possible weakness of *linguistic* PHMMs. Their reasoning is mainly due to cases like the ones displayed in (1).

```
(1)   d u e –
      d o – s
```

To produce such an alignment, it is necessary to have a transition from one gap state to the other. Another difference between the biological and the linguistic PHMM is the split of the parameter for the transition into the end state. Whilst the original version only has one parameter for this purpose, the linguistic PHMM makes use of two differ-

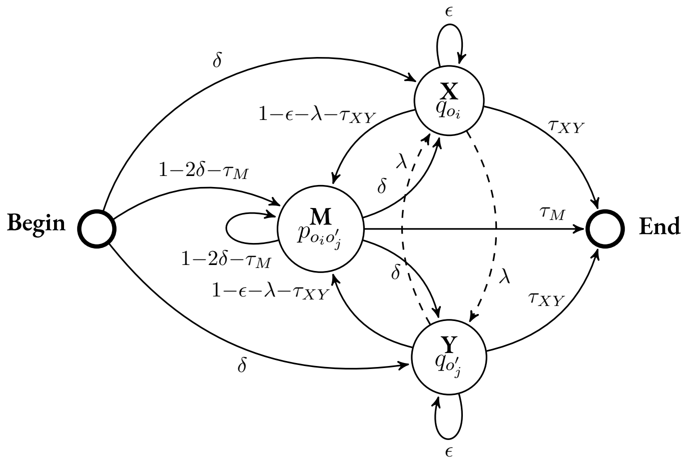


Figure 4.2: Pair Hidden Markov model as proposed by Mackay and Kondrak (2005). The states of the model are depicted by the circles and the arrows show all the possible transitions between the states. $\delta, \epsilon, \lambda, \tau_M, \tau_{XY}$ represent the transition probabilities.

ent probabilities τ_M and τ_{XY}. This split of parameters enables the model to distinguish between the match state (**M**) being the final emitting state or any of the gap states (**X**,**Y**) (see figure 4.2). This modification preserves the symmetry of the model while allowing a little bit more freedom. The different algorithms, such as the forward or Viterbi algorithm, all carry over to the *linguistic* PHMM. In order to estimate the parameters of the model, the PHMM is trained using Baum-Welch expectation maximization algorithm (Baum, 1972). To find the best alignment between two sequences x and y the Viterbi algorithm can be used. This is completely parallel to the case for DNA sequences in computational biology (see section 3.2.2). The probability of relatedness can furthermore be determined by comparing the likelihoods of the two sequences evolving under a model of relatedness (the PHMM model) or under a random model. The random model for the linguistic use case is the same as for the biological use case (cf. Mackay and Kondrak, 2005). Hence, the probability of two sequences, or rather words here, x

and y of lengths m and n, respectively which evolve independently under a null model R is given by the following equation 4.3 (repeating equation 3.21).

$$P(x,y|R) = \iota^2(1-\iota)^{n+m}\prod_{i=1}^{n} f_{x_i}\prod_{j=1}^{m} f_{y_j}, \tag{4.3}$$

with f_{x_i} being the equilibrium frequency of the sound at position i in sequence x, where $\iota = \frac{1}{\frac{m+n}{2}+1}$ (cf. Borodovsky and Ekisheva, 2006). Thus, the probability of relatedness between x and y is computed as the logarithmic ratio of the probability scores $P(x,y|\mu)$ and $P(x,y|R)$, where μ is the trained model and R is the null model. As for the PMI weighted alignment, the similarity scores obtained from the PHMM are also converted to distance scores using the sigmoid transformation.

4.3.2 Online EM

The Expectation Maximization algorithm (EM) is widely used in computational linguistics for the purpose of training models of word alignment, document classification and word segmentation. The EM algorithm starts with an initial setting of model parameters and uses these model parameters to calculate the so called *sufficient statistics*. These sufficient statistics are, for example, the number of times a state emits a symbol or a certain transition is made (3.11 and 3.12 are examples of sufficient statistics). Using these sufficient statistics, the model parameters are reestimated. The EM algorithm reestimates the model parameters after each full scan of the training data.

Liang and Klein (2009) observe that described training procedure can lead to slow convergence. As a matter of fact, Jäger (2013) trains his PMI system using the standard EM (also known as batch EM) which updates the parameters in a PMI scoring matrix only after aligning all the word pairs. In contrast, Online EM (Liang and Klein, 2009) updates the model parameters after passing through just a subset of the training data (also known as minibatch in online learning literature). The Online EM algorithm combines the sufficient statistics (s) from the current update step k with the previous parameters θ_{k-1} to calculate the new parameters θ_k using the following equation:

$$\theta_k = (1-\eta_k)\theta_{k-1} + \eta_k s \tag{4.4}$$

where η_k is defined as: $\eta_k = (k+2)^{-\alpha}$.[4] The parameter η_k determines how fast to forget or remember the updates from the previous steps. The parameter α is in the range of $0.5 \leq \alpha \leq 1$. A smaller α implies a large update to the model parameters. The parameter k is related to the minibatch parameter (m; $m = \lceil \frac{D}{k} \rceil$; where, D is the size of training data) and determines the number of updates to be performed. If $k = 1$, the EM algorithm is almost the standard EM algorithm, with the exception that update steps are still connected via η_k. The other extreme, that $k = D$ implies that an update to the parameters happens between each sample in the training data.

The training data in the current experiment consists of word pairs and the EM algorithms are the Baum-Welch algorithm for PHHMs and the procedure described in section 4.3.1.

4.3.3 Clustering Algorithm

Following List et al. (2016), the clustering algorithm is the InfoMap algorithm (Rosvall and Bergstrom, 2008). The InfoMap clustering method is an information theoretic approach to detect community structure within a connected network. The method uses random walks on a network as a proxy for information flow to detect communities, i.e., clusters, without the need for a threshold. A community is a group of nodes with more edges connecting the nodes within the community than connecting them with nodes outside the community (Newman and Girvan, 2004). In the given case, a community

[4]The particular choice of η_k is only bound through the following condition: $\sum_{k=0}^{\infty} \eta_k = \infty$ and $\sum_{k=0}^{\infty} \eta_k^2 < \infty$ (Liang and Klein, 2009). This particular version of η_k, following Liang and Klein (2009), ensures this condition.

refers to the words which are cognate and have higher edge weights between them. The idea behind the algorithm is that the random walk is statistically more likely to spend a long period of time within a community than switching communities due to the nature of the network.

The resulting scores of the alignment algorithms described above are stored as a pairwise distance matrix. Such a distance matrix can be thought of as a complete weighted graph. The cells of this matrix which have a score below 0.5 are deleted, which also removes this particular edge from the graph. A score which is lower than 0.5 after sigmoid transformation means that the original similarity score is below 0. In the case of PHMMs, this has a very natural consequence. Similarity scores below 0 indicate that the probability of being generated by the random model is higher than the probability of being generated through the random model. A similar point can be made for the PMI based Needleman-Wunsch alignment. The scoring scheme is designed such that the previous interpretation holds as well. The interpretation gets disturbed by the fact that a similar interpretation is not possible for the gap scores. However, to maintain comparability, we used the same cut-off for the PMI based alignment. Thus, a non-complete graph is constructed and the resulting network is supplied as an input to the InfoMap algorithm.

4.3.4 Experiments

The following section describes the experimental settings, datasets, evaluation measures, and the comparing systems Baseline, ALINE, PMI-LANG and LexStat, which we use to evaluate the performance of PMI based alignment and PHMMs trained with the online training method.

Hyperparameters of Online EM

The best values for m and α are determined by performing a grid search for m in the range of $m = 2^s$ where $s \in [5, 15]$; and, $\alpha \in [0.5, 1.0]$ with a step size of 0.05. The gap opening and gap extension penalties are set to -2.5 and -1.75 following Jäger (2013). The reason to fix these parameters is that there is no straightforward way to estimate these two parameters at the same time as the scoring scheme. The remaining parameters are all learned via the online training method.

Datasets

The aforementioned IELex database (Dyen et al., 1992)[5] and the Austronesian Vocabulary Database (Greenhill and Gray, 2009) which is a subset of the ABVD database by Greenhill et al. (2008), are not transcribed in a uniform IPA format. We cleaned both databases from any symbols which are not IPA standard and next converted into ASJP format. Additionally, we selected a random set of 100 languages from the Austronesian Data.[6] Additionally, we used some small databases with cognate judgments from Wichmann and Holman (2013) and List (2014b). The respective authors compiled cognacy wordlists for subsets of families from various scholarly sources such as comparative handbooks and historical linguistics' articles. The details for the different databases are given in table 4.2.

Evaluation Measures

We analyzed the clustering results using B-cubed F-score (Amigó et al., 2009). The B-cubed scores are defined for each word belonging to a meaning as followed. The precision for a word is defined as the ratio between the number of cognates in its cluster to the total number of words in its cluster. The recall for a word is defined as the ra-

[5] `http://ielex.mpi.nl/`, curated by Michael Dunn.

[6] LexStat takes many hours to run on a dataset of 100 languages.

Family	NOM	NOL	AveCC	AveWC
Austronesian	210	100	20.2142	4.1143
Afrasian	40	21	9.5	2.6868
Bai dialects	110	9	2.5909	6.0166
Chinese dialects	179	18	6.8771	5.2635
Huon	84	14	6.3929	2.7672
Indo-European	207	52	12.2126	7.3461
Japanese dialects	200	10	2.3	6.1373
Kadai	40	12	3.225	5.0027
Kamasau	36	8	1.6667	5.3981
Lolo-Burmese	40	15	2.625	7.3121
Mayan	100	30	8.58	6.1521
Miao-Yao	39	6	1.8974	3.9667
Mixe-Zoque	100	10	3	4.6535
Mon-Khmer	100	16	7.75	2.7956
ObUgrian	110	21	2.2	11.8162
Tujia	109	5	1.6422	3.3792

Table 4.2: This is an overview of the characteristics of the data sets. Number of languages (NOL), Number of meanings (NOM), Average number of cognate classes per meaning (AveCC) and Average number of words per cognate class (AveWC).

tio between the number of cognates in its cluster to the total number of expert labeled cognates. The B-cubed precision and recall are defined as the average of the words' precision and recall across all the clusters. Finally, the B-cubed F-score for a meaning, is computed as the harmonic mean of the average items' precision and recall. The Averaged B-cubed F-score for the whole dataset is computed as the average of the B-cubed F-scores across all the meanings.

Amigó et al. (2009) show that the B-cubed F-score satisfies four formal constraints known as cluster homogeneity, cluster completeness, rag bag (robustness to misplacement of a true singleton item) and robustness to variation in cluster size. The authors show that cluster evaluation measures based on entropy such as Mutual Information and V-measure (Rosenberg and Hirschberg, 2007) and Rand index do not satisfy the four constraints. Both Hauer and Kondrak (2011) and List et al. (2016) use B-cubed F-scores to evaluate their cognate clustering systems.

4.3.5 Comparing Systems

LDN

The Needleman-Wunsch algorithm is designed to determine the similarity of two strings. A high similarity between two strings also implies that there is a small distance between them. As Sellers (1974) shows, the structure of the Needleman-Wunsch algorithm can also be used to calculate the minimum distance. Instead of maximizing the similarity, the distance is minimized. Under a certain scoring scheme, this algorithm then calculates the Levenshtein distance (Levenshtein, 1966). The Levenshtein distance is the minimal number of operations to transform one string into another. Consider the two word "house" and "haus". Their Levenshtein distance is 2. There is the replacement of "o" with "a" and the deletion of "e". The scoring scheme for the Levenshtein distance will score identical symbols with a zero and non identical symbols with a one. By definition, the maximal Levenshtein distance between two strings is the length of the longest string. To fairly account for strings of different length, the Levenshtein distance can be divided by the length of the longer of the two strings. This operation yields the normalized Levenshtein distance (LDN). The normalized Levenshtein distance is adopted as the baseline in the current experiments.

ALINE

ALINE is a sequence alignment system designed by Kondrak (2000) for computing similarities between two words by decomposing phonemes into multivalued and binary phonetic features. Each phoneme is decomposed into multivalued features such as place and manner for consonants and height and backness for vowels. Multivalued features take values on a continuous scale ranging from $[0, 1]$ and the values represent the distance between the sources of articulation. Binary valued features consist of nasal, voicing, aspirated and retroflex.

Each feature is weighed by a *salience* value that is determined manually. The similarity score between two sequences is computed as the sum of the aligned sound segments. Following Downey et al. (2008), ALINE's similarity score s_{ab} between two words a, b is converted into a distance score based on the following formula: $1.0 - \frac{2.0 * s_{ab}}{s_{aa} + s_{bb}}$.[7]

PMI-LANG

Jäger (2013) developed a system that learns PMI sound matrices to optimize a criterion designed to optimize language relatedness. The core idea is to tie up word similarity to language similarity such that close languages like English/German tend to have more similarity than English/Hindi. The language similarity function amounts to maximizing similarity between probable cognates to learn a PMI score matrix. The training method and the system is also explained above. Jäger (2013) applies the learned PMI score matrix to infer phylogenetic trees of language families. However, the learned PMI score matrix has not been applied to cognate clustering yet.

LexStat

LexStat (List, 2012a) is part of LingPy (List and Forkel, 2016) library offering state-of-the-art alignment algorithms for aligning word pairs and clustering them into cognate sets. The work flow of LexStat proceeds through the following steps:

1. LexStat uses a hand-crafted sound segment matrix, h, to align and score the word pairs for each meaning. Let the similarity of a segment pair i, j be given as h_{ij}.
2. For each language pair, l_1, l_2 the word pairs belonging to the same meaning are aligned. The frequency of a segment pair i, j belonging to the same meaning is given as a_{ij}.

[7]In the current experiment we use the Python implementation provided by Huff and Lonsdale (2011), which is available at `https://sourceforge.net/projects/pyaline/`.

3. For l_1, l_2, the words belonging to one of the language is shuffled and realigned using Needleman-Wunsch algorithm. This procedure is repeated for all language pairs for 100 times. The average frequency of a segment pair i, j from the reshuffling step is given as e_{ij}.

4. All the parameters h, a, e are combined according to the following formula to give a new segment similarity score s_{ij}, where $w_1 + w_2 = 1$.

$$s_{ij} = 2 * w_1 log \frac{a_{ij}}{e_{ij}} + w_2 h_{ij} \tag{4.5}$$

5. The weights s_{ij} are then used to score word pairs and cluster words in a meaning.

The intuition behind step 3 is to reduce the effect of chance similarities between the sound segments that can obscure real genetic sound correspondences.[8] The word distances from all the above systems are all used as an input to InfoMap to infer cognate clusters.

4.3.6 Results

The current study consists of two separate experiments. While the evaluation and the methods are the same, they differ in their usage of slightly different training data.

Out-of-Family Training

In this experiment, we train our PHMM and PMI systems on the 40-item wordlists from the ASJP database belonging to families other than those language groups present in table 4.2. It is made sure that there is no overlap between the languages present in the test dataset and the training dataset. The training sets for both systems, PMI and PHMM, consist of a lot of probable cognates. In sum, there are $1, 151, 178$ word pairs.

[8]The code for LingPy can be obtained from `https://github.com/lingpy`. The LexStat similarity scores are converted into distance scores using the same formula as ALINE.

Family	LDN	PMI-LANG	Batch PMI	Online PMI	Batch PHMM	Online PHMM	LexStat	ALINE
Austronesian	0.7175	0.7355	0.6539	**0.7364**	0.6224	0.6709	0.7173	0.5321
Afrasian	0.7993	0.8133	0.7496	**0.8392**	0.7213	0.7044	–	0.6442
Bai dialects	0.8348	0.8766	0.8716	**0.8774**	0.8741	0.8639	0.8417	0.8462
Chinese dialects	0.7687	0.7521	0.7217	0.7803	0.7455	0.7396	**0.7815**	0.6651
Huon	0.8536	0.8556	0.7518	0.8775	0.7612	0.7437	–	0.6413
Indo-European	0.7367	0.7752	0.7337	**0.7812**	0.715	0.7126	0.7316	0.6583
Japanese dialects	0.893	0.9031	0.8943	0.9051	0.9006	**0.9083**	0.8875	0.8699
Kadai	0.7581	0.8175	0.8139	**0.8309**	0.8	0.8159	–	0.7647
Kamasau	0.9561	**0.9850**	0.9543	0.9823	0.9605	0.9674	–	0.9479
Lolo-Burmese	0.6469	0.713	0.7862	0.7805	0.7846	**0.8218**	–	0.8027
Mayan	**0.8198**	0.7798	0.6958	0.8074	0.6804	0.6797	0.7931	0.627
Miao-Yao	0.6412	0.7003	0.7679	0.7801	0.7411	0.7879	–	**0.8426**
Mixe-Zoque	0.9055	0.9149	0.8528	**0.9209**	0.8521	0.8599	0.8656	0.8298
Mon-Khmer	0.7883	0.8209	0.7054	**0.8302**	0.6921	0.7008	0.7925	0.6472
ObUgrian	0.8623	0.911	0.8987	**0.9214**	0.8951	0.8874	0.8837	0.8826
Tujia	0.8882	0.9091	0.9018	**0.9105**	0.895	0.9027	0.8905	0.8757
Average	0.8044	0.8289	0.7971	**0.8415**	0.7901	0.7955	0.8185	0.7548

Table 4.3: The B-cubed F-scores of different models of sixteen language groups. The last row reports the average of the B-cubed F-scores across all the datasets. The numbers in **bold** show the highest scores across columns.

The results of our experiments are given in table 4.3. As eluded above, the alignment scores are converted into distances using the sigmoid transformation and all edges with a distance below 0.5 were kept for the InfoMap clustering. It is expected for LexStat to perform better in the case of Chinese since LexStat handles tones internally whereas, the ASJP representation does not handle tones. In the case of online systems, we report the best results for m, α. Following List (2014a),we do not report LexStat results for the language groups which have word lists shorter than 100 meanings.

The Online PMI performs better than the rest of the systems at nine out of the sixteen families. On an average, the Online PMI system ranks the best followed by PMI-LANG and LexStat system. ALINE performs the best on Miao-Yao language group. The Online PMI system performs better than the Batch PMI on all the datasets. As expected, the LexStat system performs the best on Chinese dialect dataset. Surprisingly, the PHMM systems do not perform as well as the simpler PMI systems despite its complexity.

Greenhill (2011) applied Levenshtein distance for the classification of Austronesian languages and argued that Levenshtein distance does not perform well at the task of detecting language relationships. In contrast, our experiment shows that Levenshtein

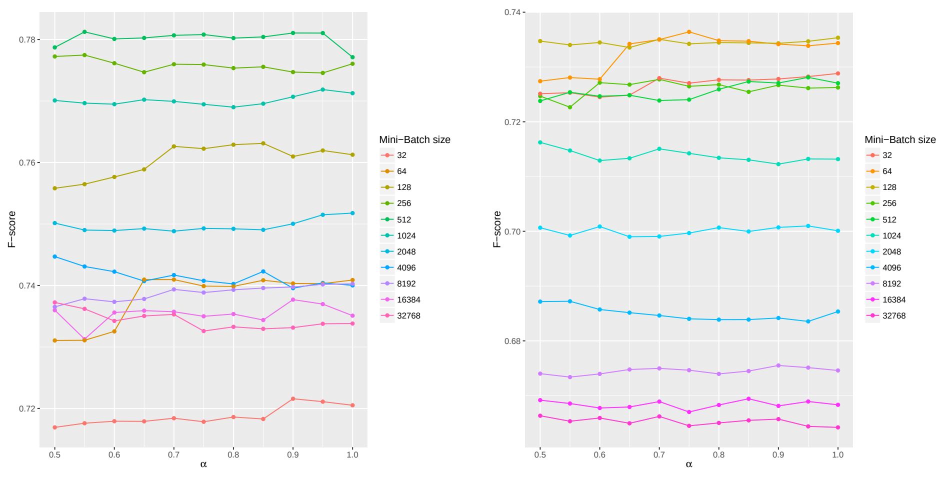
	PMI		PHMM	
Family	m	α	m	α
Austronesian	64	0.75	32	0.5
Afrasian	256	0.65	32	0.8
Bai dialects	8192	0.75	32	0.55
Chinese dialects	128	0.95	512	0.6
Huon	32	1	32	0.65
Indo-European	512	0.55	1024	0.5
Japanese dialects	512	0.55	32	0.6
Kadai	2048	0.7	32	0.7
Kamasau	512	0.5	128	0.55
Lolo-Burmese	16384	0.5	32	0.75
Mayan	64	0.5	32	0.55
Miao-Yao	8192	0.95	128	0.7
Mixe-Zoque	256	0.7	32	0.7
Mon-Khmer	256	0.7	32	0.5
ObUgrian	512	0.75	32768	0.5
Tujia	1024	0.65	32	0.5

Table 4.4: Best settings of m and α for Online variants of PMI and PHMM

distance comes close to LexStat in the case of Austronesian language family. Both PMI-LANG and Online PMI are two points better than Levenshtein distance at the task of cognate identification.

The results are much clearer in the case of Indo-European language family. The PMI-LANG and Online PMI systems perform better than the rest of the systems. Levenshtein distance performs better than LexStat for the Indo-European language family. On an average, ALINE shows the lowest performance of all the systems.

Table 4.4 shows the corresponding setting of m, α for all the online systems. The value of m is quite variable across language families whereas α tends to be in the range of $0.5 - 0.75$. The effect of m and α for Indo-European and Austronesian languages are plotted against the results of the Online PMI system in figures 4.3 (page 78). The B-cubed F-scores are stable across the range of α but show variable results for value of m. The top-3 F-scores for Indo-European are at $m = 256, 512, 1024$ and at $m = 64, 128, 256$ for Austronesian language family. These results suggest that the online



(a) Indo-European **(b)** Austronesian

Figure 4.3: Plots of m and α against B-cubed F-scores for out-of-family training

Family	Training word pairs	Online PHMM			Online PMI			Batch PHMM	Batch PMI
		m	α	F-score	m	α	F-score		
ASJP Indo-European	380769	128	0.60	0.7646	4096	0.60	**0.7868**	0.7656	0.7704
Indo-European	25386	64	0.50	0.7901	1024	0.85	**0.7971**	0.7797	0.7914
ASJP Mayan	91665	256	0.55	0.7765	128	0.90	**0.8250**	0.7814	0.7677
Mayan	11889	32	0.55	0.7952	64	0.70	**0.7997**	0.7888	0.7544
ASJP Austronesian	1000000	32	0.65	0.6190	128	0.80	**0.7453**	0.6239	0.6429
Austronesian	84311	32	0.5	0.6709	128	0.80	**0.7460**	0.6517	0.6509

Table 4.5: The results of training the PMI and PHMM systems on the ASJP 40 word lists and the full word lists of Indo-European, Mayan and Austronesian.

training helps cognate clustering more than the batch training. The plots shown in figure 4.3 suggest that small batch size improves the performance whereas a large batch size (eg., 32, 768) hurts the performance on Indo-European and Austronesian language families.

Within-Family Training

In this experiment, the PMI and PHMM systems are trained on the three largest language families in the dataset: the Mayan, Indo-European, and Austronesian language families. The training data for this experiment comes from another source than the test data. We perform experiments in two different settings.

1. The ASJP database has 40-length word lists for more languages ($\sim$ 3 times) than the languages in cognate databases of Mayan, Indo-European, and Austronesian language families. The database allows the access of more word pairs than any other database in existence.

2. A list of probable cognate pairs from the IELex, ABVD, and Mayan language databases are extracted.

The motivation behind these experiments is to investigate the performance of both PMI and PHMM systems when trained on the word lists belonging to the same language family but compiled by different groups of annotators. A successful experiment indicates that this approach of training a PMI matrix on ASJP 40 word lists can be applied to language families that have longer word lists but no cognate judgments. The number of training word pairs and the results of our experiments are given in table 4.5 on page 78.

The Online variants perform better than the batch systems across all language families and settings. Online PMI performs better across all the language families than the Batch PMI. Online PMI trained on ASJP word lists of a language family show close performance to an Online PMI system trained within the language family in the case of Indo-European and Austronesian language families. The performance of batch PMI system comes close to the Online PMI system in the case of Indo-European but falls behind in the case of other language families. Training the online system on ASJP word lists improves the performance in the case of the Mayan language family. This performance is not observed in the case of Indo-European and Austronesian language families. The reason for this could be due to the source of the origin of the datasets.

The batch PMI and PHMM systems perform better than the LexStat method on the Indo-European and the Mayan language family. The Online PHMM system comes close in performance to the Online PMI system in the case of Indo-European and Mayan language families. PHMM systems show the lowest performance on Austronesian language family. Except for Indo-European, the best batch sizes for online PMI system are small and are typically ≤ 256.

4.3.7 Discussion

Effect of m and α

Throughout the experiments, one can observe that low minibatch size gives better results than a large minibatch size. Another observation is that an intermediary value of α is usually sufficient for obtaining the best results. Figure 4.3 on page 78 shows that small values of m yield stable F-scores across the range of α. Small values of m typically give better results than larger values of α. In contrast to other NLP tasks that require a large m and smaller α, the task of aligning two words requires smaller values of m. The small value of m implies a large number of updates which is important for a task where the average sequence length (~ 5) and the average number of word pairs are smaller than $100,000$. Further, an intermediary value of α controls the amount of memory retained at each update.

Speed

One advantage of our online systems (either PMI or PHMM) is that the training time typically lasts around ten minutes on a single thread of an i7-6700 processor. In the case of PHMM, online training speeds up the convergence and overall yields better results than the batch variant. In comparison, the PMI-LANG system takes days to train. Finally, the results show that the online algorithm can yield better performance than LexStat. LexStat and PHMM take more than 5 hours to test on the language subset of the Austronesian language family. In contrast, PMI (both online and batch) takes less than ten minutes for each value of m, α in the case of out-of-family training. Usually, five scans over the full data were sufficient for convergence.

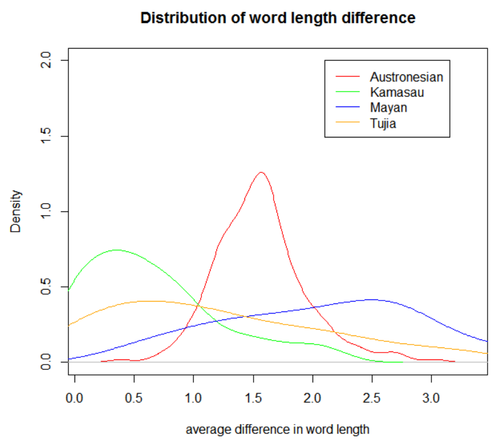


Figure 4.4: Distribution of the average difference of word lengths across concepts

Analyzing PHMM's Performance

Although PHMMs are the most complex among the tested models, the performance of these models is not as good as the conceptually simpler PMI models. This lack of performance could be due to the characteristics of the PHMM. The transition probability from the *begin* state to the *match* or *gap* states is the same as the transition probability from the match state to either gap state or itself (figure 4.2, p 67). Although desirable for biological purposes, this poses a big problem for linguistic applications. To start an alignment with a match is more likely than to start it with a gap.[9] Therefore, the alignments generated by PHHMs are more likely to show gaps at the end of the string than in the beginning. This results in problems for data sets where word length differs considerably. The PHMM performs the worst for those datasets that show a huge difference in the word length. On the other hand, for Kamasau and Tujia – the two datasets with the best performance – the difference in word length is much less pronounced (cf. figure 4.4). Based on the results of these experiments, training the PMI-based segment scores in an online fashion and supplied to InfoMap clustering, could yield reliable cognate judgments.

[9] $1 - 2\delta\tau_M$ is larger than δ in all models (cf. figure 4.2, p 67).

Future work could attempt to solve the problem by reformulating the transition probabilities between the states, especially the transitions out of the begin state. In its current formulation, these transitions are set to the same value as their corresponding transition out of the match state. This parameterization still stems from the origins of the PHMM in computational biology. In addition, assigning different probability values to the transitions from the states **X** and **Y** to the end state may make the model even more suitable for linguistic applications.

4.3.8 Conclusion

In this study, we evaluated the performance of various sequence alignment algorithms – both learned and linguistically designed – for the task of cognate detection across different language families. Training PMI and PHMM in an online fashion speeds up convergence and yields comparable or better results than the batch variant and the state-of-the-art LexStat system. Online PMI system shows the best performance across different language families. In conclusion, PMI systems can be trained faster in an online fashion and yield better accuracies than the current state-of-the-art systems.

It is a well known point in historical linguistics that the similarity between languages depends on regular sound changes which have happened between them. It remains an open question how to include this into pairwise alignment algorithms and further improve their performance. Notably, Hruschka et al. (2015) developed a statistical model to distinguish regular sound changes from irregular ones. However, their approach is based on aligned data. This shifts the focus of attention to detecting regular sound changes rather than using information about regular sound changes to further improve the quality of alignments and cognate detection.

4.4 Phylogenetic Inference from automatic cognate judgments

As historical linguistics typically looks at the identification of cognates and language phylogenies as a very intertwined problem, a lot of studies solely focus on either of the tasks. This division of labor is typical for the area of computational historical linguistics. One of the goals of computational historical linguistics, however, is to automatically infer the language tree from the raw data. When using the cognacy information obtained from automatic methods as an input for phylogenetic inference, algorithms need to be tested. Rama et al. (2018) used such automatically inferred cognates for the inference of language trees and compared these trees to gold standard data. The aim of this study is to identify how useful automatic cognate judgments are already in the downstream task of inferring language trees. I am going to present this study in this section in more detail, since it is an attempt to further automate the process of the classical comparative method.

Rama et al. (2018) tested six different automatic methods of cognacy identification. Of these six, four are already discussed above: Online PMI, LexStat, LDN and the SVM method of Jäger et al. (2017). The two additional methods are the Consonant-Class-Matching (CCM) method by Turchin et al. (2010) and the Sound-Class-Based alignment (SCA) by List (2012c, 2014b).

4.4.1 Two Additional Approaches

The study by Rama et al. (2018) uses a slightly different set of cognate identification methods than the one presented above. In addition to LexStat, LDN and the Online PMI method, the authors added two approaches to the toolbox which heavily lean on sound classes (Consonant-Class-Matching and Sound-Class-Based alignment) as well as a supervised approach based on a Support Vector Machine (SVM). In the following, I will shortly introduce these three additional methods.

Consonant-Class-Matching

The CCM approach to cognate identification proceeds in a rather rigorous manner. In a first step, all consonants are mapped onto one of eight consonant classes and in a second step, all vowels are deleted. Cognacy is then determined on the basis of the first two consonant classes present in each word. If the first two consonant classes for a pair in question are identical the words are judged as cognate and not cognate otherwise. The reasoning behind this method is, that words which begin with consonants sharing the same sound class are likely to go back to a common ancestor. To illustrate the approach, consider the example in (2). The German and the English words, respectively, are supposed to be cognate. (3) shows the respective translation into the consonant classes proposed by Turchin et al. (2010).

(2) a. Zahn vs. tooth
    b. Vater vs. father

(3) a. CN vs. TS
    b. PTR vs. PTR

While the first two consonant classes match in (3)-b and the two words are correctly identified as cognates, the same method fails in (3)-a.[10] An advantage of this method is the low number of false positive cognate judgments.

Sound-Class-Based alignment

The SCA method (List, 2012c, 2014b) proposed an idea which also makes use of the sound classes.[11] However, this approach is less strict than the CCM approach. There are 28 sound classes in total for the SCA approach. These classes are inspired by Dolgopolsky sound classes (Dolgopolsky, 1964, 1986). The sounds of the words are mapped

[10]List (2014b) discusses similar examples and points out a way to optimize the approach.

[11]See also section 5.1 for an overview of the sound classes.

Method	Austro-Asiatic	Austronesian	Indo-European	Pama-Nyungan	Sino-Tibetan
CCM	0.71	0.7	0.75	0.74	0.48
LDN	0.73	0.77	0.69	0.53	0.49
SCA	0.76	0.78	0.81	0.71	0.56
LexStat	0.76	0.84	0.83	0.84	0.6
OnlinePMI	0.76	0.81	0.82	0.72	0.56
SVM	0.82	0.81	0.79	0.86	0.5

Table 4.6: B-cubed F-scores for different cognate detection methods across the language families

to their respective classes. Based on a linguistically informed scoring scheme, the two words are aligned using a dynamic programming paradigm similar to the Needleman-Wunsch algorithm. All wordpairs whose score is above a certain threshold are regarded as cognate. Following List et al. (2017), Rama et al. (2018) set the threshold to 0.45.

Support Vector Machines for Cognate identification

Jäger et al. (2017) trained a linear SVM to distinguish between cognate and non-cognate word pairs. They used PMI similarity (Jäger, 2013) and LexStat distances among others as predictors. The trained SVM computes the probability of two words being cognate. These scores then serve as input for the InfoMap clustering algorithm.

Comparing the methods

Although several of the used cognate clustering tools were already used and compared above, table 4.6 shows how the additional cognate detection approaches applied by Rama et al. (2018) line up against the ones already known. Rama et al. evaluate the quality of the cognate sets inferred by the above described methods using the B-cubed F-score (Amigó et al., 2009) (see section 4.3.4 above). The authors note that the superior performance of th LexStat Algorithm compared to the SVM system for Austronesian, Indo-European and Sino-Tibetean is somewhat surprising. Since the LexStat scores function as a predictor in the SVM system, they expected SVM to outperform LexStat on the cognate detection task. In contrast, the difference in performance between OnlinePMI and SCA compared to simpler systems such as CCM and LDN is plausible. The author's hypothesis was that the higher the F-score the better the quality of the inferred phylogenetic trees.

The SVM system achieved the best scores for Austro-Asiatic and Pama-Nyungan whereas LexStat algorithm performs the best in the case of rest of the datasets. This is surprising since LexStat scores are used as features for SVM. Generally, it would have been expected that the SVM system performs better than LexStat in all the language families. Though, both OnlinePMI and SCA systems perform better than the algorithmically simpler systems such as CCM and LDN. Given these F-scores, the author's hypothesis is that the cognate sets output from the best cognate identification systems would also yield the high quality phylogenetic trees.

4.4.2 Bayesian Phylogenetic Inference

Using the MrBayes (Huelsenbeck and Ronquist, 2001; Ronquist and Huelsenbeck, 2003) software package, Rama et al. (2018) compute a posterior distribution of phylogenetic trees. This kind of software was, as previously described, developed to infer phylogenetic trees from biological data (cf. section 3.3.2). However, these software packages can also be used for the linguistic case.[12] Such a distribution is a result of Bayesian phylogenetic inference.[13] As the name suggests, this inference mechanism uses Bayes rule. Equation 4.6 presents a version of Bayes formula, already expressed in a way suitable for the current study.

$$f(\tau, v, \theta|X) = \frac{f(X|\tau, v, \theta)f(\tau, v, \theta)}{f(X)} \tag{4.6}$$

where X is the data matrix, τ is the topology of the tree, v is the vector of branch lengths and θ is the substitution model parameters. The data matrix X is a binary matrix of

[12] Yanovich (2017) gives and introduction to the usage of these models for the purpose of linguistics.
[13] See section 5.3 for a more detailed introduction to these methods.

dimensions $N \times C$, where N is the number of languages and C is the number of cognate clusters in a language family. It is difficult to calculate an analytic solution to the posterior distribution $f(\tau, v, \theta|X)$. One would need to sum over all possible topologies of the tree to compute the marginal in the denominator.[14] Using the Markov Chain Monte Carlo (MCMC) method, the posterior can be calculated without resorting to the evaluation of the denominator. The software used in this experiment uses the Metropolis-Hastings (MH) algorithm as its MCMC algorithm to sample phylogenies from the posterior distribution (Huelsenbeck et al., 2001). The posterior gets estimated by constructing a Markov Chain over the states of the parameters. Step by step, a new value for a parameter, or a block of them, gets proposed and accepted with a probability r. Let θ be the current value and θ^* be the proposed value drawn from a distribution $q(\theta^*|\theta)$.

$$r = \frac{f(X|\tau, v, \theta^*)}{f(X|\tau, v, \theta)} \frac{f(\theta^*)}{f(\theta)} \frac{q(\theta|\theta^*)}{q(\theta^*|\theta)} \tag{4.7}$$

The likelihood of the data $f(X||\tau, v, \theta)$ is computed using the Felsenstein's pruning algorithm (Felsenstein, 1981), also known as the sum-product algorithm (Jordan, 2004) (see section 3.5). In line with previous research, Rama et al. (2018) assume independence between τ, θ, v.

4.4.3 Experiments

Rama et al. (2018) binarize the cognate sets which were estimated from the different cognate identification methods such that 1 indicates the presence of a cognate class and 0 the absence. For the details of the model definition, see Rama et al. (2018). In order to measure the quality of the inferred trees, they calculate the Generalized Quartet Distance between each tree in the posterior and the gold standard tree. This distribu-

[14]The number of tree topologies would already be huge for small N. For N languages, there are $\frac{(2N-3)!}{2^{N-2}(N-2)!}$ different topologies.

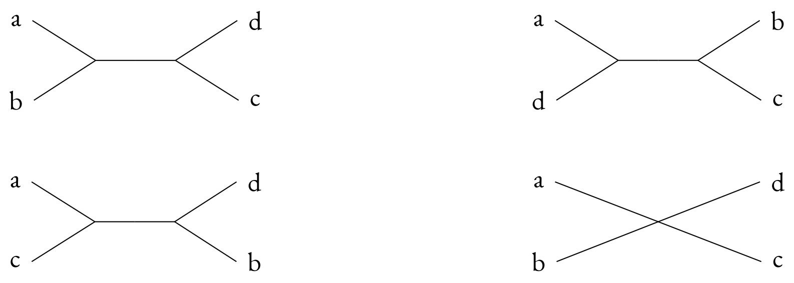


Figure 4.5: The four different topological arrangements of a quartet. The bottom right constellation is called a star and the others are called butterflies.

tion gives an estimate on how reliably a phylogenetic tree can be reconstructed from the automatically inferred cognate sets.

Generalized Quartet Distance

The Generalized Quartet Distance (GQD) (Pompei et al., 2011) improves on the Quartet Distance (QD) by enabling a comparison of binary and polytomous trees. While the trees generated by the MrBayes are all binary, gold standard trees which were taken from Glottolog (Hammarström et al., 2017) are usually not, i.e., they have non-binary internal nodes.

QD and GQD compute the distance between two trees by measuring the number of different quartets (Estabrook et al., 1985). A quartet is defined as a set of four leaves which are taken from a tree without replacement. Thus, for a tree with n leaves, there are $\binom{n}{4}$ quartets in total. Figure 4.5 shows the different arrangements of four leave nodes. In order to calculate the QD, the shared number of butterflies and stars of a pair of trees is counted and then divided by the total number of quartets. For two trees τ and τ_g, the QD is then defined as $1 - \frac{|S(\tau) \cap S(\tau_g)| + |B(\tau) \cap B(\tau_g)|}{\binom{n}{4}}$, where $S(\tau)$ counts the number of stars in τ and $B(\tau)$ the number of butterflies in τ. Since the inferred trees are necessarily resolved, i.e., there are no stars in an inferred tree, the QD counts butterflies in these trees as an error if they are stars in the gold standard tree. To salvage this problem, Pompei et al. (2011) define GQD such that a star in the gold standard tree dos not

Method	Austro-Asiatic	Austronesian	Indo-European	Pama-Nyungan	Sino-Tibetan
Expert cognate sets	**0.0081 ± 0.001**	0.1056 ± 0.0118	**0.0249 ± 0.0079**	**0.1384 ± 0.0225**	**0.0561 ± 0.0123**
CCM	0.0243 ± 0.018	0.0854 ± 0.0176	0.0369 ± 0.0148	0.1617 ± 0.0162	0.1424 ± 0.027
LDN	0.0265 ± 0.007	0.0458 ± 0.0152	0.046 ± 0.0132	0.196 ± 0.0166	0.1614 ± 0.0282
SCA	0.0152 ± 0.0035	0.0514 ± 0.013	0.0256 ± 0.009	0.166 ± 0.0153	0.0704 ± 0.0206
LexStat	0.0267 ± 0.0085	0.0848 ± 0.0226	0.0314 ± 0.0091	0.1507 ± 0.0143	0.0786 ± 0.0209
OnlinePMI	0.0158 ± 0.0048	0.1056 ± 0.0198	0.0457 ± 0.0135	0.1717 ± 0.0185	0.1184 ± 0.031
SVM	0.0146 ± 0.0039	0.0989 ± 0.0224	0.0452 ± 0.011	0.1827 ± 0.0237	0.1199 ± 0.0269

Table 4.7: Results of the phylogenetic inference algorithms. The mean and standard deviation for each method and family is computed from 7500 posterior trees. The automatic methods which come closest to the gold standard phylogeny is shaded in grey, and where the expert cognate sets perform best, this is indicated with a **bold** font.

negatively influence the score. The GQD counts the number of different butterflies in both trees and divides it by the resolved butterflies in the gold standard tree. It follows that a higher GQD score indicates higher dissimilarity between trees. The GQD is used for comparing inferred language phylogenies with gold standard phylogenies (Greenhill et al., 2010; Wichmann et al., 2011; Jäger, 2013).

4.4.4 Results

Rama et al. report that with the exemption of Austronesian, the trees generated from inferred cognate data resemble the gold standard tree fairly close. Table 4.7 shows the results of Rama et al. (2018). Somewhat unexpectedly, algorithmically simpler systems such as LDN or CCM outperform more complex systems such as the SVM model on almost all data sets. This trend also holds for LexStat and SCA, where the latter is a subsystem of the former. However, the authors summarize that in the absence of expert-coded cognate data, trees generated from automatically inferred data can function as a reasonable approximation.

5 Cognacy Estimation under a Time-Dependent Alignment Model

Previous approaches to alignment and cognacy estimation do not employ the idea of evolutionary time (cf. chapter 4). The alignment model which I propose here uses the idea of evolutionary time for alignment and cognacy estimation. Including the notion of evolutionary time enables adjustments of parameters for a given pair of languages based on the time separating the two. This idea is different form language specific parameters which are already used by the LexStat system (List, 2012a). While the LexStat approach merges a set of global parameters with language specific parameters, this approach uses the toolbox which comes with models of DNA evolution (cf. section 3.3). Thus, there is a Q-matrix of sound substitution rates from which I calculate a language specific set of substitution probabilities.

Generally, there are two components to an alignment model. One set of parameters specifies the substitution parameters for the symbols, e.g., nucleotides or sounds, and another set defines the transition probabilities between substitution or indel events. Section 5.1 shows a parameterization of a model of "sound evolution", which I use to derive the actual sound substitution probabilities. The TKF models (Thorne et al., 1991, 1992) are models where state transition parameters are sensitive to evolutionary time. Therefore, these models lend themselves naturally for a time dependent alignment model.

These two ingredients can be combined to formulate a time-dependent model for the calculation of alignment and cognacy using the notion of a tree. Since the branch length of already existing trees might not necessarily correspond to the time scale of these models, it is necessary to infer the branch lengths of the trees underlying the model of evolutionary time. I develop a likelihood function for this model in section 5.2. Parameter inference is done via a Markov Chain Monte Carlo (MCMC) approach which I describe in section 5.3. Section 5.4 explains how cognacy estimation is performed under this model. Since the parameters are estimated using an MCMC algorithm, the result is a posterior distribution of parameters. This distribution can be used to derive a posterior distribution over cognate judgments instead of an absolute judgment as it is the case for other approaches (section 5.4). To evaluate the performance of this new model, a standard cognate inference task is performed and evaluated. Section 5.5 describes the experimental settings and the evaluation.

5.1 Q-matrices for Sounds

Developing a Q-matrix for linguistic purposes is pretty straightforward in itself. Instead of a 4×4 matrix as in the case of DNA sequences, it will be a $n \times n$ matrix, where n is the size of the alphabet used in the experiments[1]. However, this results in an explosion

[1]For the experiments presented here $n = 39$

of parameters for the linguistic case. There are $n - 1$ parameters for the equilibrium frequencies and $\frac{n(n-1)}{2}$ parameters for the evolutionary rate values. Although the parameters should be constrained such that the expected amount of change per unit in time is one, these are quite a lot of parameters to estimate. Thus, it is desirable to reduce the parameter space. Reducing the frequency parameters by assigning the same frequency value to different, probably related sounds, does not reduce the parameter space significantly since $\frac{n(n-1)}{2} > n$. Therefore, reducing the number of evolutionary rate parameters is much more useful. Fortunately, the idea of sound classes can be leveraged here. The idea of using sound classes is not new to computational historical linguistics. List (2012c) uses the idea of Dolgopolsky classes (Dolgopolsky, 1964, 1986) to align words and detect cognates. The ten classes originally proposed by Dolgopolsky are shown in table 5.1 (p. 93). The SCA approach (List, 2012c) extends the idea of sound classes to a total of 28 sound classes. In comparison to the original model, which was focused on the Indo-European languages, the SCA model is also capable of handling tone.

The sound-class based approach to alignment encodes the sounds of a word as their respective sound class. In the model proposed by Dolgopolsky (1986), aligning different sound classes is forbidden. The approach taken by the SCA-method relaxes this assumption and introduces a scoring scheme to code probabilities of divergent matches. These probabilities are estimated based on a weighted directed graph. The directions of the edges mirror the direction of a change and the weights mirror the respective probability. A small weight indicates a high probability of change. Based on such a graph, List (2012c) calculates a scoring scheme.

The course I take here does not try to recode sounds as sound classes but rather weaves the idea into the Q-matrix. The idea is to assign the same evolutionary rate to sounds from the same sound class. To illustrate the approach, assume there are only five sounds in the alphabet $\Sigma = \{$ s, p, b, t , d$\}$ and they are assigned to the sound

class	description	examples
P	labial obstruents	p,b,f
T	dental obstruents	d,t,θ,ð
S	sibilants	s,z,ʃ,ʒ
K	velar obstruents, dental and alveolar fricatives	k,g,ts,tʃ
M	labial nasal	m
N	remaining nasals	n,ŋ
R	liquids	r,l
W	voiced labial fricative and initial rounded vowels	v,u
J	palatal approximant	j
∅	laryngeals and initial velar nasal	h,ɦ

Table 5.1: Dolgopolsky's sound classes (Dolgopolsky, 1986) as provided by List (2012b)

classes listed in table 5.1. A Q-matrix in the style of the General Time Reversible model (Lanave et al., 1984) is shown in 5.1.

$$Q = \begin{pmatrix} \cdot & \alpha\pi_p & \beta\pi_b & \gamma\pi_t & \delta\pi_d \\ \alpha\pi_s & \cdot & \epsilon\pi_b & \zeta\pi_t & \eta\pi_d \\ \beta\pi_s & \epsilon\pi_p & \cdot & \theta\pi_t & \iota\pi_d \\ \gamma\pi_s & \zeta\pi_p & \theta\pi_b & \cdot & \kappa\pi_d \\ \delta\pi_s & \eta\pi_p & \iota\pi_b & \kappa\pi_t & \cdot \end{pmatrix} \tag{5.1}$$

In the standard way of assigning evolutionary rates, there is a different rate for every cell in the upper triangle of the matrix. The sound-class-based approach taken here assigns the same evolutionary rate ϵ to within sound class matches across all classes but assigns different rates for between class matches. (e.g., ζ or γ).

$$Q' = \begin{pmatrix} \cdot & \alpha\pi_p & \alpha\pi_b & \gamma\pi_t & \gamma\pi_d \\ \alpha\pi_s & \cdot & \epsilon\pi_b & \zeta\pi_t & \zeta\pi_d \\ \alpha\pi_s & \epsilon\pi_p & \cdot & \zeta\pi_t & \zeta\pi_d \\ \gamma\pi_s & \zeta\pi_p & \zeta\pi_b & \cdot & \epsilon\pi_d \\ \gamma\pi_s & \zeta\pi_p & \zeta\pi_b & \epsilon\pi_t & \cdot \end{pmatrix} \tag{5.2}$$

This approach reduces the number of parameters considerably. While there were 741 evolutionary rate parameters with an alphabet size of 39, there are now 56.[2] The amount of parameters can be reduced even further by assigining the same evolutionary rate parameter to the match between any consonantal sound class and the vocalic sound class. An exploratory analysis, which allowed for different parameters between the consonantal classes and the vocalic class, converged on the same value for all these parameters. Thus, using the reduction in the parameter space is supported from the data and theoretical considerations. While maintaining the interpretation that matches of sounds between classes share certain characteristics, either by having the same value as in List (2012c) or Dolgopolsky (1986) or by sharing a certain amount of probability mass, this approach reduces the complexity of the model significantly.

The formulation of the Q-matrix shown above also bears some similarity with other models of DNA evolution. The model proposed by Tamura and Nei (1993) implements the idea that changes of nucleic acids are of one of two kinds, *transitions* or *transversions*. While *transitions* are changes within the same group, *transversions* are changes between groups. The idea of changes within and between sound classes, as used here, runs very much in parallel. However, the model by Tamura and Nei (1993) places some very strong restrictions on the frequency parameters. The proposed Q-matrix for

[2]The experiments reported here assume the 11 Dolgopolsky classes proposed by List (2012c).

sounds is therefore not a direct extension of the Tamura and Nei model, but rather a mix of the General Time reversible model, in that it allows varying base frequencies, and the Tamura and Nei model in that it uses the idea of groups.

5.2 Building the Model

Before further developing the model, it is necessary to discuss a key difference to statistical alignment models in computational biology. What separates this model from its biological counterparts is the basic assumption of homology. Statistical alignment models and almost all other models in computational biology assume at least functional equivalence or even homology of all the sequences under consideration. The assumption is that all sequences are related under some model of evolution for which the parameters can be inferred. This picture changes for computational historical linguistics. Especially in the realm of alignment and cognacy detection, the task is to decide whether two sequences are related or not. For this reason, homology cannot be assumed a priori for linguistic applications. For this reason, existing software from computational biology such as StatAlign (Novák et al., 2008; Arunapuram et al., 2013) or BaliPhy (Redelings and Suchard, 2005; Suchard and Redelings, 2006) cannot be used for this task in computational historical linguistics. Since the inference of cognate classes is part of the task here, supplying this information beforehand is unwarranted. In order to assess the applicability of time dependent alignment models for CHL, I choose a fairly straightforward approach.

Throughout the remainder of this chapter, $s_a^{c_1}$ denotes the sequence of language A describing concept c_1. In this fashion, languages are modelled via a set of sequences. Thus, $s_a^{c_1}$ and $s_b^{c_1}$ denote two sequences describing the same concept, one from language A and one from language B. The notation s_a, s_b will be used as a shorthand for $s_a^{c_i}, s_b^{c_j}$, where $i = j$, i.e., these sequences are semantically aligned. The superscript will also be dropped if a possible ambiguity is resolved by the context. Additionally, $t_{a,b}$ is the length of the path connecting node A and node B in a (phylogenetic) tree τ.

Since the time dependent alignment model is only sensibly defined for homologous sequences, any likelihood function for a model tasked with the identification of cognates needs to account for this. The general idea is the following: If two sequences are related via common ancestry, i.e., they are cognate, there is an evolutionary process which generated the sequences. I describe this process using the time dependent alignment model. If two sequences are not cognate (not related via common ancestry) there is no such process and these sequences evolved independently. Independent evolution is modelled similarly to the random model described for PHMMs (cf. Figure 3.5 on page 34). These considerations lead to the following definition:

$$Pr[s_a, s_b|\theta, t_{a,b}] = \max\left[Vit(s_a, s_b, \theta, t) \cdot e^{-t_{a,b}},\ R(s_b, s_a, \theta, t) \cdot 1 - e^{-t_{a,b}}\right] \tag{5.3}$$

Where $Vit(s_a, s_b, \theta, t)$ is the score calculated by the Viterbi algorithm (cf. section 3.2.2), θ are the necessary parameters, i.e., substitution matrix, transition probabilities etc., $t_{a,b}$ is the evolutionary time separating s_a and s_b. $R(s_b, s_a, \theta, t)$ is the score calculated under a model of unrelated evolution. In terms of mere cognate detection, there is no apparent a-priori reason to prefer the Viterbi algorithm over the forward algorithm in the definition of the likelihood function. However, from a modelling point of view the usage of the Viterbi algorithm shifts the model more into the direction of a maximum-likelihood approach than a Bayesian model. Yet, I test both algorithms in the experiments below to explore the capabilities of the time-dependent alignment model. Given the TKF model and the model of sound evolution described above, calculating the likelihood of two sequences under unrelated evolution is straightforward. The TKF model assumes that the lengths of sequences are distributed according to a geometric distribution (equation 3.23 is repeated here as 5.4), where λ is the birth and μ the death rate.

$$P(length(s_a) = n) = \left(1 - \frac{\lambda}{\mu}\right)\left(\frac{\lambda}{\mu}\right)^n \tag{5.4}$$

The actual elements of the sequences, i.e., the sounds of the words, are distributed according to their equilibrium frequencies π. The likelihood of two sequences being generated independently is then defined as the product of the likelihood of the two processes.

$$\begin{aligned} R(s_a, s_b|\lambda, \mu, \pi) &= P(length(s_a) = n) \cdot P(length(s_b) = m) \\ &= \left(\frac{\lambda}{\mu}\right)^{n+m}\left(1 - \frac{\lambda}{\mu}\right)^2 \prod_{i=1}^{n} \pi_{a_i} \prod_{j=1}^{m} \pi_{b_j}. \end{aligned} \tag{5.5}$$

The score calculated under the random model and the Viterbi score are weighted by $1-e^{-t}$ and e^{-t}, respectively. The intuition is the larger the time separating the two, the less probable is cognacy.[3] It is important to note that this definition of the model does not use any a-priori information about potential relatedness between the sequences. As illustrated above, the Viterbi algorithm needs as its parameters the equilibrium frequencies, a matrix of sound substitution probabilities and state transition probabilities. The state transition probabilities are calculated according to the TKF model (see section 3.2.3). The matrix of sound substitution probabilities is derived from a Q-Matrix

[3]This idea is related to the ideas proposed in Glottochronology Lees (1953).

which is constructed as described in section 5.1. This process is named *Q-Func* in the following. Putting these different parts together results in the following model.

$$\begin{array}{lcl}
s_a, s_b & \sim & Pr[s_a, s_b | t_{a,b}, \pi, exp(Q \cdot t_{a,b}), \mu, \lambda] \\
\pi & \sim & Dirichlet(|\Sigma|, \alpha) \\
\alpha & \sim & Exponential(1) \\
t_{a,b} & = & PathLength(A, B, \tau) \\
\tau & \sim & CompoundDirichlet(1, 1, 0.1, 1) \\
r_k & \sim & LogNormal(\kappa, \sigma) \\
\sigma & \sim & Exponential(1) \\
\kappa & \sim & Normal(0, 0.02) \\
Q & = & \textit{Q-Func}(r, \pi) \\
\lambda & \sim & Exponential(1) \\
\mu & \sim & Exponential(1)
\end{array} \tag{5.6}$$

The Compound Dirichlet distribution (Zhang et al., 2012) places a rather widespread prior on the length of a tree and then partitions this length using a Dirichlet distribution. Importantly, the only aspect of τ that is sampled is the branch lengths.

There are two aspects of the definition of the model which deserve some discussion. The first aspect is the prior distribution on the evolutionary rates. By definition, the rates need to be positive which eliminates all distributions defined on negative numbers. The *LogNormal* distribution with a mean value close to 0 is peaked at 1. In order to remain neutral about the relative weights of the evolutionary rates a-priori, a prior distribution peaked at 1 is a good fit. This is exactly what the *LogNormal* distribution can do.

The second aspect is concerned with the parameters μ and λ. The TKF91 model requires λ to be smaller than μ. This would turn the priors of the two parameters to two truncated exponential distributions which change with each change in μ and λ.

To solve this problem, the model is reparametrized. Let $\rho = \frac{\lambda}{\mu}$, since $\mu > \lambda$ and both being positive the following condition holds.

$$0 < \rho < 1 \tag{5.7}$$

Placing a prior which is defined on the interval between 0 and 1 on ρ and formulating μ in terms of ρ and λ removes the complex relations of the priors. This second consideration changes the model definition slightly to the following.

$$\begin{array}{lcl}
s_a, s_b & \sim & Pr[s_a, s_b | t_{a,b}, \pi, exp(Q \cdot t_{a,b}), \mu, \lambda] \\
\pi & \sim & Dirichlet(\alpha) \\
\alpha & \sim & Exponential(1) \\
t_{a,b} & = & PathLength(A, B, \tau) \\
\tau & \sim & CompoundDirichlet(1, 1, 0.1, 1) \\
r_k & \sim & LogNormal(\kappa, \sigma) \\
\sigma & \sim & Exponential(1) \\
\kappa & \sim & Normal(0, 0.02) \\
Q & = & \textit{Q-Func}(r, \pi) \\
\mu & = & \frac{\lambda}{\rho} \\
\lambda & \sim & Exponential(1) \\
\rho & \sim & \frac{1}{1+\rho'} \\
\rho' & \sim & Exponential(1)
\end{array} \tag{5.8}$$

5.3 Parameter Estimation

The last section described a method to calculate the likelihood of a given time dependent alignment model. As a next step, it is necessary to estimate the parameters of the model. The approach I take here is to find a distribution of parameters which describes the model, i.e., the distribution $P(\theta|D)$ needs to be estimated. According to Bayes' formula this quantity can be calculated as follows:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}. \tag{5.9}$$

From this equation we know that the probability of the model parameters given the data, i.e., $P(\theta|D)$, can be calculated by multiplying the likelihood $P(D|\theta)$ with the prior $P(\theta)$ and by dividing this product by the marginal likelihood of the data $P(D)$. The calculation of the likelihood can be carried out using the method described above in section 5.2. The prior can be calculated by carefully selecting the appropriate distribution. This makes the numerator easy to calculate. However, determining the marginal likelihood of the data turns out to be difficult. In theory, this quantity can be calculated by integrating over all possible parameter values as well.

$$P(D) = \int_\theta P(D|\theta)P(\theta)d\theta \tag{5.10}$$

Although this integration can be broken down into a summation over topologies (τ) and a multidimensional integration over parameter values for the substitution model, the transition model and the branch lengths (χ), it is impossible to calculate the value of the integral directly (Ronquist et al., 2009).

$$P(D) = \sum_\tau \int_\chi P(D|\tau, \chi)P(\chi)d\chi \tag{5.11}$$

Since this marginal probability cannot be calculated analytically, calculating the posterior distribution $P(\theta|D)$ analytically is not possible either. However, by using the Markov Chain Monte Carlo (MCMC) method, it is still possible to get an estimate of the posterior distribution. The idea is to build a Markov chain of the parameters which converges to an equilibrium state which resembles the posterior distribution. The equilibrium state is also independent of the starting point of the chain. Constructing such a chain can be achieved by using the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970). The general idea of this method is to propose a new state, i.e., make a change to the parameters and accept it if the likelihood increases or reject it with some probability if the likelihood does not increase. The steps of the Metropolis hastings algorithm are shown below.

1. Start with a random parameter vector θ,
2. propose a new state θ',
3. calculate the ratio r of the posterior probabilities of θ' and θ
   (a) if $r > 1$ accept θ'
   (b) else accept θ' with probability r, if θ' is rejected stay in θ,
4. return to step 2.

The change to this algorithm which Hastings (1970) proposed was in the calculation of r. In the original formulation by Metropolis et al. (1953) all parameter changes are equally likely. Hastings (1970) adjusted the method such that parameter changes can be weighted by their probability. Using this extension, the acceptance probability can be calculated as follows.

$$\begin{aligned} r = &\min\left(1, \frac{P(\theta'|D)}{P(\theta|D)} \times \frac{P(\theta|\theta')}{P(\theta'|\theta)}\right) \\ &\min\left(1, \frac{\frac{P(\theta')P(D|\theta')}{P(D)}}{\frac{P(\theta)P(D|\theta)}{P(D)}} \times \frac{P(\theta|\theta')}{P(\theta'|\theta)}\right) \\ &\min\left(1, \frac{P(\theta')}{P(\theta)} \times \frac{P(D|\theta')}{P(D|\theta)} \times \frac{P(\theta|\theta')}{P(\theta'|\theta)}\right) \end{aligned} \tag{5.12}$$

The last term in the third row of equation 5.12 accounts for different probabilities of parameter moves. In the formulation by Metropolis et al. (1953) all parameter moves are equally likely, i.e., $P(\theta'|\theta) \equiv P(\theta|\theta')$, which means that this probability cancels out. Another important aspect is that the intractable marginal probability cancels out. This makes r easy to compute. Furthermore, the construction of r ensures that after a sufficient amount of time, the proportion of time a chain spends sampling a particular parameter is proportional to the posterior probability of that parameter (Ronquist et al., 2009). Due to the fact that the first parameter estimate is random, the likelihood in the beginning is usually low. For the chain to properly sample from the posterior region with a high probability mass without being influenced by the initial choice of model parameters, an initial amount of samples will be discarded. This phase of the sampling procedure is called the *burnin*. When the chain starts to sample from the equilibrium distribution, the likelihood seems to converge onto a plateau. Since the Metropolis-Hastings algorithm proposes a change based on the current value of a parameter, successive values are highly correlated. In order to get independent samples, usually only every n-th sample is recorded. This is also known as *thinning*.

5.3.1 Improved Sampling

The traditional Metropolis-Hastings algorithm is known to be rather inefficient in its sampling behavior. This has lead to the development of several techniques which lead to a faster convergence of the algorithm (Andrieu and Thoms, 2008). In order to speed up and improve the sampling behavior for the present experiment, I use a mixture of proposal moves. More technically, the transition matrix of a Markov chain can be interpreted as a transition kernel in continuous space. It can be shown that a mixture of such kernels where each kernel has the same invariant distribution $p(\cdot)$ also has the invariant distribution $p(\cdot)$. Thus it is possible to use different samplers for the parameters to achieve more efficient sampling behavior (Andrieu et al., 2003). The topology of the tree is sampled using traditional random-walk Metropolis-Hastings. For the other parameters a slice sampler is used. These parameters are the branch lengths of the tree,

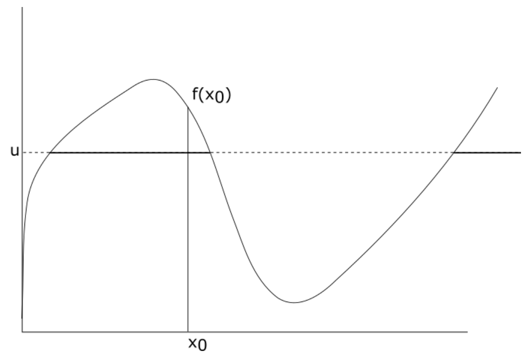


Figure 5.1: Illustration of the slice sampler. A vertical level is drawn between 0 and $f(x_0)$ to define a horizontal slice.

equilibrium frequencies of the sounds, the evolutionary rate parameters and the parameters of the transition model.

Slice Sampling

The idea of the slice sampling (Damlen et al., 1999; Higdon, 1998) approach is to sample from an augmented distribution rather than from the original one. The reason for this approach is that it is often more convenient to sample from the distribution $p^*(x, u)$ than it is to sample from $p(x)$. (Andrieu et al., 2003; Neal, 2003) This new extended distribution $p^*(x, u)$ is defined as follows:

$$p^*(x,u) = \begin{cases} 1 & \text{if } 0 \leq u \leq p(x) \\ 0 & \text{else} \end{cases} \tag{5.13}$$

Sampling x then reduces to jointly sample from (x, u) and then to ignore u. Figure 5.1 helps to illustrate the procedure of the slice sampling approach.

1. A horizontal slice is defined (indicated by the continuous line) by drawing a vertical line in the interval between 0 and $f(x_0)$.

2. A window around x_0 is drawn from which the new point x_1 is sampled.

3. If x_1 is inside the slice, i.e., equation 5.13 is 1, it is accepted, otherwise a new point is drawn and the window will be shrunk.

There are several techniques on how to select and adapt the proper window around x_0 (steps 2 and 3). Neal (2003) presents some algorithms for selecting and adapting the window. The slice sampling approach can also be used to sample from a multivariate distribution.

In case a vector valued parameter $v = (v_1, v_2, \ldots, v_n)$ is such that for all v_i in v it is the case that $0 < v_i < 1$ and $\sum_{i=1}^{n} x_i = 1$, the support for the posterior distribution of this vector is the standard (n-1) simplex (Cowles et al., 2009).Sampling from this simplex can be done using an algorithm presented by Cowles et al. (2009). In the current experiment, the equilibrium frequencies as well as the evolutionary rates meet the constraints for the simplex sampling approach. I sample the values for the branch lengths and the transition model using the standard slice sampling scheme. It is important to note that sampling the branch lengths vector does not change the topology of the tree. The set of branch lengths associated with one particular tree topology form a euclidean space (Dinh et al., 2017). Thus, slice sampling of a particular branch length vector is possible.

5.4 Cognacy Estimation in the Time-Dependent Model

Evaluating the time dependent alignment model is not as straightforward as it is for other alignment approaches. The time-dependent alignment model does not define a distribution over cognate classes. Thus, measuring the models ability to correctly predict cognate classes can only be evaluated indirectly. The approach taken here is as follows: By using a set of parameters from a particular draw from the posterior, a traditional pairwise cognacy detection task can be performed, resulting in a cognate/non-cognate statement for each pair of semantically aligned words. The same steps are carried out for each draw . These statements can be counted and the proportion of the respective cognate/non-cognate statements are calculated. This results in an approximation of a posterior distribution over cognate classes instead of an absolute statement as it is done in other approaches.

The task is to label each target word pair as cognate or non-cognate for a given set of parameters. Using the alignment model described above, the probability of relatedness for a target pair can be calculated. However, similar to other approaches, it is necessary to determine a threshold to judge the pair as cognate or non-cognate.[4] The approach taken here is partly inspired by Jäger (2013) and Hein et al. (2000).[5] Although Jäger (2013) did not focus on the identification of cognate word pairs, there are some similarities in the approach. Jäger (2013) observes that the pairwise distances of related words are much lower than for non-cognate word pairs. Following up on this observation, the scores of all word-pairs (potential cognates and definite non-cognates) are ranked. The resulting rankings show an expected pattern: For unrelated languages, the distribution of ranks approximate a uniform distribution, while for related languages, the distribution is heavily skewed towards the small values. He continues to use this skewness to derive a distance measure which is based on the shape of the distribution mentioned before. In the current approach the idea of using the scores of unrelated words, and comparing them to the scores of related words is developed into a threshold method for cognacy detection.

The idea is as follows: The alignment score of related sequences should be considerably different from alignment scores of random sequences. Although the parameterization of this random distribution is hard to obtain analytically, an approximation of this distribution given the data is possible. For a given pair of languages and a Swadesh list of words for each language, imagine a grid with the words belonging to the first language along the x-axis and the words of the other language listed along the y-axis (cf. table 5.2 on page 106).

[4]See section 5.7.1 for a discussion of alternatives.

[5]Similar suggestions were already made in the pioneering paper by Needleman and Wunsch (1970).

German / English	Ich	Du	Blut	Laus
I	-2.16	-6.16	-9.60	-6.53
You	-5.63	0.98	-2.65	-2.65
blood	-7.31	-3.33	6.01	-2.82
louse	-7.52	-0.07	0.90	12.22

Table 5.2: Alignment scores using ASJP encoding the alignment mechanism described by Jäger (2018).

The simplifying assumption made here and also in other research dealing with the identification of cognates is to assume that potential pairs of cognates can only appear along the diagonal of the data structure shown above. All the pairs on the off-diagonal are supposed to reflect random word pairs. Thus, the alignment scores in the off-diagonal cells can be used to approximate a random distribution. The decision if a given word pair is judged as cognate or not comes down to comparing a particular alignment score to the distribution of random scores. The remaining question is: "How big should the difference between random and potential cognates be?" The "difference from randomness" is parameterized as being above the nth-percentile of the random distribution. This question can either be solved theoretically or by a grid search optimizing the percentile cut-off. Independent of the choice of the exact cut-off, each pair above the cut-off is considered to be cognate or non-cognate otherwise. This results in a statement of cognacy for each word pair.

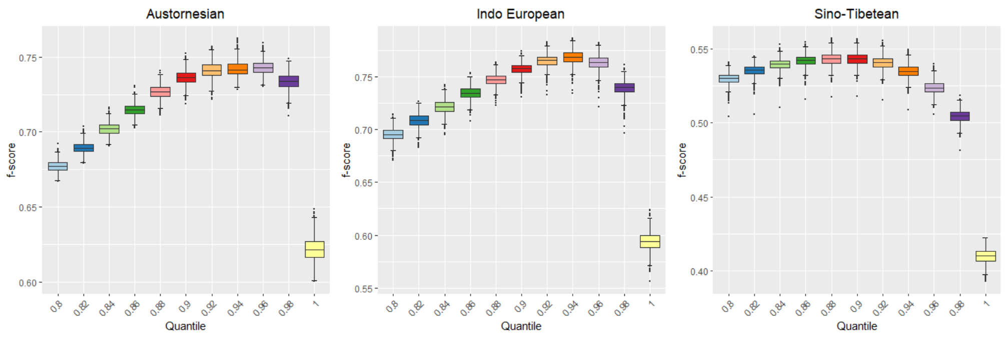
In the description of the model above, cognate word pairs are aligned using the Viterbi algorithm in the first and the forward algorithm in the second experiment. Crucially, in comparison to other approaches, these are the plain algorithms and not the respective log-odds variants. The log-odds variants of the Viterbi or forward algorithm are used to correct for sequence length or chance similarities. The models formulated above do not use the log-odds variants. This is due to the characteristics of the TKF 91 model and the explicit modelling of sound evolution. The TKF91 model is based on an clearly specified model of sequence length and includes this resulting distribution in the formulation of its transition probabilities. Thus, a correction for the different probabilities of sequence lengths is already a part of the model and need not to be added. This also holds for the model of sound evolution. This separates the current model from other approaches such as the PMI approach. In contrast to those models, where a correction for spurious sound similarity is necessary since mere sound frequencies are used for the similarity scores, the model used in this thesis solves this problem conceptually. For these reasons, the cognate identification is done on the plain variant of the Viterbi and forward algorithm rather than with their respective log-odds variants.

5.5 Experiments

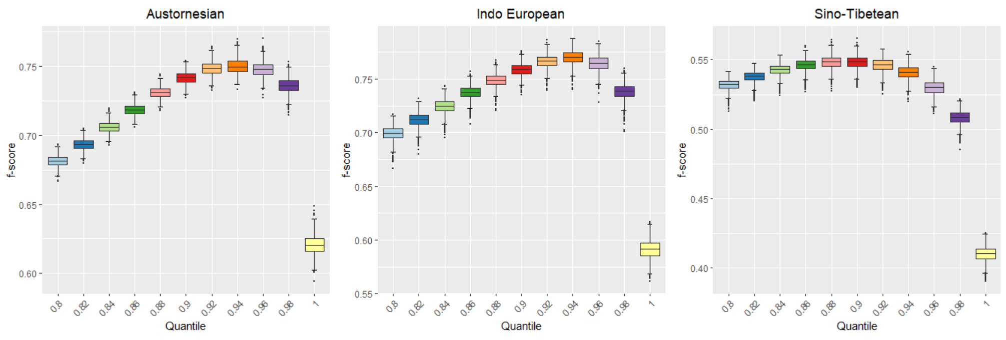
I evaluate, the evolutionary alignment model on a standard cognate identification task. Each sample from the posterior distribution functions as a set of parameters for the kind of cognate identification procedure described above. Using the parameters from each sample from the posterior, each wordpair can be judged as cognate or not. For each sample there is a matrix of cognate judgements similar to the experiment described in the previous chapter. These matrices are used as an input to the InfoMap algorithm (Rosvall and Bergstrom, 2008). The resulting clusters are evaluated using B-cubed F-scores. The experiments are performed on the three datasets Indo-European, Austronesian and Sino-Tibetean , which were also used in the study by Rama et al. (2018).

5.6 Results

I perform a grid search among quantiles as described in the previous section; The resulting posterior distribution of F-scores are plotted in figure 5.2 (see page 108). The plots all show a similar pattern, namely the highest median F-score is achieved by treating a wordpair as cognate if the alignment score is above the 90th to 94th quantile. Although all results are numerically different, this pattern holds across datasets and algorithms.



(a) Posterior distribution of B-cubed F-scores for the model using the Viterbi algorithm.



(b) Posterior distribution of B-cubed F-scores for the model using the forward algorithm.

Figure 5.2: Resulting distributions of the posterior B-cubed F-scores for the two different model variants evaluated on the three different datasets. The x-axes show the different quantiles and the y-axes the respective F-scores.

	Viterbi		forward	
	F-score	Quantile	F-score	Quantile
Austronesian	0.743	0.96	0.750	0.94
Indo European	0.769	0.94	0.770	0.94
Sino-Tibetean	0.543	0.88	0.548	0.88

Table 5.3: Highest median B-cubed F-score values for the different experimental settings and the corresponding quantile.

	Viterbi			forward		
	F-score	Quantile	Cut-off	F-score	Quantile	Cut-off
Austronesian	0.757	0.84	0.95	0.761	0.88	0.90
Indo European	0.78	0.94	0.55	0.783	0.92	0.80
Sino-Tibetean	0.55	0.82	0.95	0.555	0.82	0.70

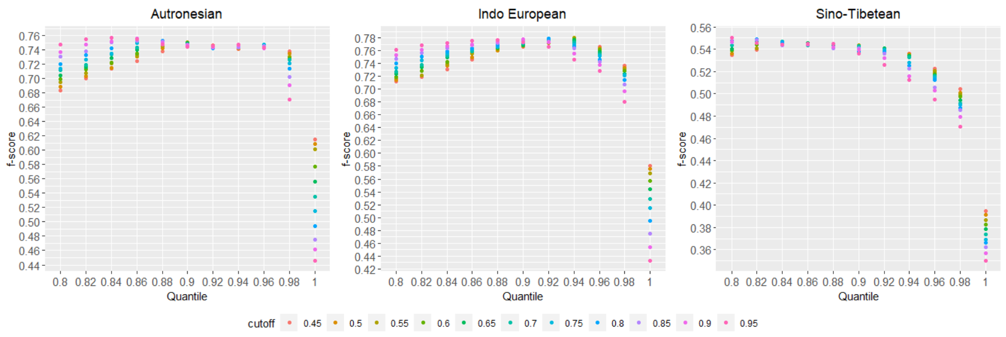
Table 5.4: Optimal F-score values with corresponding quantile and cut-off values for the pairwise probability evaluation method.

Table 5.3 (p. 108) lists the highest median B-cubed F-score values with the corresponding quantile. This method of evaluating the model provides a posterior distribution over cognate clusters.

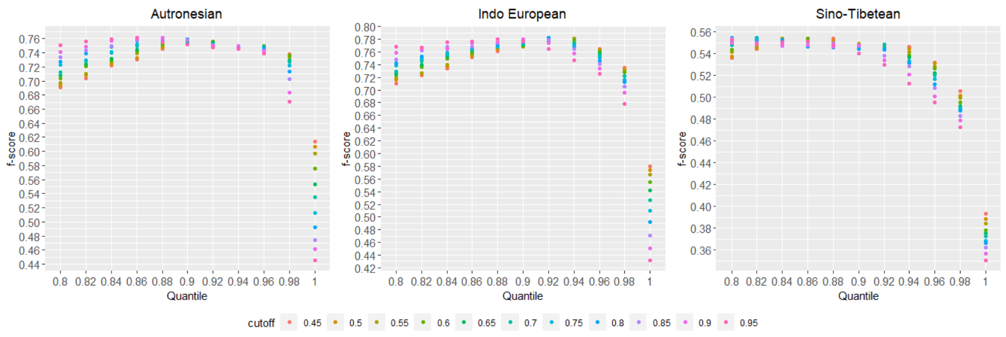
Another way of evaluating the experiments is to get a posterior point probability of each individual wordpair being cognate. Instead of estimating a cognate cluster for each sample from the posterior, the pairwise judgements are aggregated over the full posterior. This results in a probability of each wordpair being cognate. This method of evaluating the experiments is again performed as a grid search among quantiles. In addition a similar grid search for the cutoff parameter in the infomap-clustering algorithm is performed (cf. section 4.3.3). This results in singular F-scores instead of a distribution. Figure 5.3 on page 110 gives an overview of the different results; table 5.4 lists the highest F-scores and their corresponding cutoff and quantile settings. Regarding the quantile value, the results of this evaluation technique do not show a trend similar to the previous one. Generally speaking, wordpairs above the 88th to 94th quantile of all alignment scores for a pair of languages tend to correspond to actual cognate words. Unfortunately, a similarly clear statement is not possible regarding the cutoff value for the infomap-algorithm.

5.7 Discussion

I tested two different evaluation methods. In both approaches, I used samples from the posterior distribution to generate pairwise cognate judgements. For the first eval-



(a) B-cubed F-scores for the pairwise probability evaluation technique of the model using the Viterbi algorithm.



(b) B-cubed F-scores for the pairwise probability evaluation technique of the model using the forward algorithm.

Figure 5.3: Results of the pairwise probability evaluation technique for the two different model variants evaluated on the three different datasets. The x-axes show the different quantiles and the y-axes the respective F-scores.

uation, I inferred a distribution over cognate classes with these judgements. For the second evaluation, I aggregated the pairwise judgements and based on this aggregation inferred cognate classes. Comparing the two evaluation methods, it can be stated that in terms of point estimates, the overall better performance is achieved using the second evaluation technique. This behaviour stems from the more nuanced judgements of similarity which are supplied to the clustering algorithm. While the similarity scores used for the clustering in the first technique are either 1 or 0, the scores in the second technique are not binary but continuous. The infomap-algorithm is actually capable of picking up the differences of the continuous values. This property of the clustering algorithm is lost for the first technique.

In comparison to the other cognate detection techniques presented in chapter 4.4, it has to be stated though that the results of both methods are rather in the middle of the pack than actual frontrunners. In its general architecture, the time dependent alignment model is build on the foundation of a PHMM. The PMI approach was already superior to the PHMM model setup in the experiments presented in section 4.3.6.

5.7.1 Comments

Although I formulated the model described in this chapter as a hierarchical Bayesian model, care has to be taken. I evaluated the model on a cognate identification task by generating distribution of cognate judgements and cognate classes. This should not be confused with an actual posterior distribution over cognate classes or judgements. The only thing the model can provide in its current form is a posterior distribution over alignment scores. These scores can be used in a cognate identification task. Hence, the evaluation of the model is done indirectly.

A fully Bayesian model should integrate the cognate classes into its likelihood function. Such a model could be used to estimate an actual posterior distribution over cognate classes. Thus, the model described above should not be confused with a model which samples cognate classes in a completely Bayesian fashion. Additionally, as already noted above, the usage of the Viterbi algorithm moves the model towards a maximum-likelihood model. This model can, however, be taken as a first step into the direction of a Bayesian modelling of cognacy. It shows that an explicit model of sound evolution can be employed for computational historical linguistics, which might be a crucial component in a full fledged Bayesian model for cognate classes.

There is an alternative to the way to assess cognacy. In the definition of the likelihood function, the result of the $\max$ operator can be probed to get an estimate of the judgements of the model. If the larger value is the one stemming from the related model, the respective pair of words is judged cognate or not cognate. Such an approach would eliminate the comparison of the diagonal vs. off-diagonal scores. However, a tentative evaluation suggest sub-par performance in terms of cognate detection. In addition, this way of setting-up the model has the same shortcomings discussed in the previous paragraph in that it is still not fully Bayesian.

5.8 Conclusion

This chapter shows how a hierarchical Bayesian model can be defined for the task of cognate identification. Using an explicit model of sequence and sound evolution, the model includes phylogenetic information, or rather the branch lengths of the pyhlogenetic tree, to use this information for cognate detection. While the results do not match current state of the art methods yet, this model has the potential to be improved through its modular nature. In comparison to other existing methods for cognate detection, this is the first Bayesian approach to this task. By directly including a phylogenetic tree into the cognate detection paradigm, this model can be seen as a step towards merging cognate identification and phylogenetic inference into one unified model.

6
No-U-Turn Sampling for Phylogenetic Trees

As already mentioned above, some samplers used for Markov Chain Monte Carlo methods tend to have a slow mixing behavior and are thus rather inefficient. Especially the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970), which has been the workhorse for phylogenetic research, is prone to slow mixing. In recent years, several different sampling techniques were proposed which can be used to improve the sampling procedure. Such a technique is the slice sampling approach presented above (see section 5.3).[1] While these improved methods are able to guide the random walk of the original Metropolis-Hastings procedure, they still show a local random walk behavior. In order to avoid such a random walk, Hamiltonian Dynamics can be used to create

[1] Andrieu et al. (2003) provide an overview of different variants on how to improve MCMC sampling behavior.

another type of Monte Carlo samplers (Duane et al., 1987). Borrowing from physics, Hamiltonian Monte Carlo enriches the parameters of the model by corresponding 'momentum' variables to rapidly explore the target distribution and suppress the local random walk (Gelman et al., 2013).

In section 6.1, I will shortly introduce the idea of Hamiltonian dynamics and their application in MCMC settings, especially how it is used in the No-U-Turn sampler (Hoffman and Gelman, 2014). Section 6.2 is concerned with the problems of applying the concept of Hamiltonian dynamics to the space of trees and a solution to this problem. In section 6.3 I develop a No-U-Turn Sampler for the space of trees. Section 6.4 presents some experiments using the Phylogenetic-No-U-Turn Sampler.

6.1 Hamiltonian Dynamics and the No-U-Turn Sampler

Hamiltonian dynamics provide a mathematical framework to model the dynamics of a physical system over time using the systems energy. Neal (2011) provides the following example.

> In two dimensions, we can visualize the dynamics as that of a frictionless puck that slides over a surface of varying height. The state of the system consists of the position of the puck, given by a 2D vector q, and the *momentum* of the puck (its mass times its velocity) given by a 2D vector p. The *potential energy*, $U(q)$, of the puck is proportional to the height of the surface at its current position, and its *kinetic energy*, $K(p)$, is equal to $|p|^2/(2m)$, where m is the mass of the puck (Neal, 2011, p. 2).

The momentum of the puck will help it to slide along a rising slope as long as the kinetic energy is above zero. At the point where the kinetic energy becomes zero it will naturally slide down again. In a non physical setting the position of the puck is equated to the variable of interest and the potential energy to the minus of the log probability density of the given variable. Only the momentum variable will be set artificially (Neal, 2011). By using the Hamiltonian equations, the development of p and the parameters θ over time t can be determined:

$$\begin{aligned} \frac{d\theta_i}{dt} &= \frac{\partial H}{\partial p_i} \\ \frac{dp_i}{dt} &= \frac{\partial H}{\partial \theta_i} \\ i &= 1 \ldots d. \end{aligned} \tag{6.1}$$

Where H is the Hamiltonian function taking p and θ as arguments and d the dimension of p and θ. The function $H(\theta, p)$ can be written as

$$H(\theta, p) = U(\theta) + K(p). \tag{6.2}$$

It has been shown in other places (e.g., Duane et al., 1987; Neal, 2011) that Hamiltonian dynamics exhibit the mathematical properties which are necessary to generate valid Markov Chain Monte Carlo updates.

6.1.1 Leapfrog approximation

For a practical implementation of Hamiltonian Dynamics, the continuous equations in the system need to be approximated. This can be solved by using a discretization approximation with a small stepsize ε. The state of the system is then computed at intervals of ε starting at time zero. The method which may first come to mind for such an approximation is Euler's method. However, another method has prevailed in practical applications. The method of choice is the *leapfrog* integrator, which is a slight modification of Euler's method. Euler's method does a small step ε for the momentum and the position (i.e., parameter) vector in turn. In contrast, the leapfrog integrator first does a half step of the momentum variable, then full step for the position variables, i.e., the variable which is sampled, and then again a half step for the momentum variable (Neal, 2011).

$$\begin{aligned} p_i\left(t+\frac{\epsilon}{2}\right) &= p_i\left(t\right) - \left(\frac{\varepsilon}{2}\right)\frac{\partial U}{\partial \theta_i}\left(\theta\left(t\right)\right) \\ \theta_i(t+\epsilon) &= \theta_i\left(t\right) + \varepsilon\frac{p_i\left(t+\frac{\varepsilon}{2}\right)}{m_i} \\ p_i(t+\epsilon) &= p_i\left(t+\frac{\varepsilon}{2}\right) - \left(\frac{\varepsilon}{2}\right)\frac{\partial U}{\partial \theta_i}\left(\theta\left(t+\varepsilon\right)\right) \end{aligned} \tag{6.3}$$

In one update step, the leapfrog approximation is done for L steps. This describes the Hamiltonian Monte Carlo (HMC) algorithm. The discretization procedure necessarily introduces a slight error into the computation. The error depends on ε and L and can theoretically grow out of bounds depending on L (Leimkuhler and Reich, 2005). In real world applications, however, this is usually not an issue (Hoffman and Gelman, 2014). The leapfrog integrator fulfills all properties which are necessary for valid updates on the parameters (see Neal (2011) for formal details). Algorithm 1 on page 117 shows the general procedure of the Hamiltonian Monte Carlo method including the leapfrog approximation.

What remains for the user to do is to specify L and ε. In practice, setting these two parameters can be tedious. As Hoffman and Gelman (2014) point out, an ε which is too small will waste precious computation time while a ε which is too large will produce inaccurate samples. Similarly, if L is too small, the HMC mirrors random walk behavior and if L is too large the HMC will start to loop back and retrace its steps. In some severe cases, if L is too large, some of the formal properties making the leapfrog method a valid sampler may even break down (Neal, 2011). Thus fine-tuning L and ε is very important.

Algorithm 1 The pseudo code for HMC following Hoffman and Gelman (2014). With $\mathcal{L}$ being the logarithm of the joint density of the variable θ.

```
Given $\theta^0$, $\varepsilon$, $L$, $\mathcal{L}$, $M$
for $i = 1$ to $M$ do
    Sample $p^0 \sim \mathcal{N}(0, I)$
    Set $\theta^i \leftarrow \theta^{i-1}, \tilde{\theta} \leftarrow \theta^{i-1}, \tilde{p} \leftarrow p^0$
    for j=1 to $L$ do
        Set $\tilde{\theta}, \tilde{p} \leftarrow$ LeapFrog($\tilde{\theta}$,$\tilde{p}$, $\varepsilon$)
    end for
    With probability $\alpha = min\left\{1, \frac{exp\{\mathcal{L}(\tilde{\theta}) - \frac{1}{2}\tilde{p}\cdot\tilde{p}\}}{exp\{\mathcal{L}(\theta^{i-1}) - \frac{1}{2}p^0\cdot p^0\}}\right\}$, set $\theta^i \leftarrow \tilde{\theta}, p^i \leftarrow -\tilde{p}$
end for

function LeapFrog($\theta, p, \varepsilon$)
    Set $\tilde{p} \leftarrow p + (\varepsilon/2)\nabla_\theta\mathcal{L}(\theta)$
    Set $\tilde{\theta} \leftarrow \theta + \varepsilon\tilde{p}$
    Set $\tilde{p} \leftarrow \tilde{p} + (\varepsilon/2)\nabla_\theta\mathcal{L}(\tilde{\theta})$
    return $\tilde{\theta}, \tilde{p}$
end function
```

6.1.2 No-U-Turn sampler

To avoid tedious and time consuming fine tuning of L and ε in several trial experiments, Hoffman and Gelman (2014) developed the No-U-Turn sampler (NUTS). This sampling technique extends HMC by adaptively setting the number of steps L and optimizing ε during the burnin phase.

NUTS uses a stochastic optimization technique to optimize ε during the burnin (Andrieu and Thoms, 2008; Hoffman and Gelman, 2014). The starting value for ε is set by either doubling or halving ε until the probability of accepting an HMC update with $L = 1$ crosses 0.5.[2] Usually the seed value for ε is 1. By simulating an acceptance probability of the proposed HMC moves during burnin, ε is altered such that the overall acceptance probability approximates a given value. Hoffman and Gelman (2014) suggest a target acceptance probability between 0.45 and 0.65.

[2]A HMC update with $L = 1$ is also known as a Langevin proposal.

In contrast to the original HMC sampling scheme where the number of leapfrog steps L is a fixed parameter, the number of steps in NUTS can change in each generation. Importantly, L is not set randomly but rather such that it maximizes the distance between successive proposals while also avoiding a loop back. Furthermore, varying the number of steps accounts for the shape of the probability density surface in the current region. The algorithm proceeds by building a set of candidate states following the slice sampling approach, i.e., only if a proposed state is within the slice, it is added to the set. This set building procedure continues until one of two stopping criteria is reached. The two criteria which are checked are whether the proposed states start to retrace themselves or if the sampling error introduced via the leapfrog discretization becomes too large. As soon as the extension of the candidate set is terminated, a new state is randomly selected from the set. This way, a maximally distant new proposal state can be ensured while avoiding sampling errors and loop back behavior. There are two points within this design of the algorithm which warrant some attention. These are the stopping rules and the formation of the set of new states.

The construction of the candidate set proceeds by simulating the Hamiltonian dynamics forward and backward in time. The final state of a forward time simulation will be called θ^+ and θ^- will be the result of a backwards time simulation. Starting from the initial state the time direction is chosen randomly. For the first iteration, the Hamiltonian dynamics are simulated for one step. For the next iteration, the dynamics are simulated for two steps and the next for four steps. Consequently, every new iteration doubles the number of leapfrog steps taken. This procedure implicitly builds a balanced binary tree with the leaves corresponding to position-momentum states (Hoffman and Gelman, 2014). More formally, let $\mathcal{C}$ be the set of candidate states and let $\mathcal{B} \supseteq \mathcal{C}$ be the set of states a leapfrog integrator visits during an iteration. Hoffman and Gelman (2014) show that, as long as the initial state of the current generation is an element of $\mathcal{C}$ and for all other elements of $\mathcal{C}$ it is the case that they are in the current slice and the stopping criteria are not met, sampling a random element from $\mathcal{C}$ are valid updates. Building up $\mathcal{C}$ in this way is only valid if the method doing so preserves volume.[3] Neal (2011) shows that the leapfrog integrator does preserve volume, so it can be used to propose new states for $\mathcal{C}$. While the algorithm for building candidate sets and sampling from it fulfills all necessary conditions of being a valid sampler and also proposes distant states from the starting state, Hoffman and Gelman (2014) modify their initial algorithm slightly to improve its memory efficiency and also enable it to do larger jumps on average. The modifications include an earlier breakout of the simulation procedure if one of the stopping criteria is met and a more efficient sampling from $\mathcal{C}$ which prevents the storing of all states in memory.

In comparison to the intricate mechanism of proposing new candidate states, the stopping rules of NUTS are rather straightforward. Recall the two tests which stop further simulation of Hamiltonian dynamics. The first checks for sampling inaccuracies and the second for an indication of retracing of states. Formulating the first condition is rather uncomplicated. Hoffman and Gelman (2014) suggest to compare the Hamiltonian to some constant Δ_{max} and check if the loss of energy is larger than this constant. The other condition checks whether the simulation of the Hamiltonian dynamics would suggest that a retracing of states is going to happen. This condition is checked via simulating the dynamics for another infinitesimal amount forward and backward. If this simulation would reduce the distance between the new states, the simulation is stopped.[4]

6.2 Hamiltonian Dynamics in the Tree Space

Hamiltonian Monte Carlo methods are defined on a Euclidean parameter space. Therefore, the application of this technique for sampling phylogenetic trees is not straightfor-

[3] See Betancourt (2018) for a conceptual overview over Hamiltonian Dynamics and volume preservation.

[4] Betancourt (2013a,b) present a generalization to the No-U-Turn measure.

ward. Trees consist of two parts, a discrete graph defining the topology and an associated continuous space of branch lengths. Importantly, Billera et al. (2001) propose a continuous formulation of the space of trees. Building on this work, Dinh et al. (2017) proposed a Hamiltonian Monte Carlo sampler for trees.

The set of all binary phylogenetic trees with N leaves will be called T_N. An element of the set T_N is a pair (τ, q), where τ defines the topology and q the vector of associated branch lengths. For a particular tree, there is a topology τ and a vector $q \in \mathbb{R}^n_{\geq 0}$, where n is the dimension of q. The individual branch lengths are bounded by 0 for technical reasons (Dinh et al., 2017).[5] For each topology τ of the trees in T_N, there is a specific space $\mathbb{R}^n_{\geq 0}$ associated with it. There is a specific case where for each topology τ, the value of each entry of q is 0. In this case all trees are indistinguishable. This point is called the origin of the space. If defined in this way, the different spaces $\mathbb{R}^n_{\geq 0}$ form so called orthants. For a set of N leaves there are $(2N - 3)!!$ such orthants. This formulation neatly gives rise to an intuitive neighboring relation between trees embedded in this space. Imagine reducing the length of an internal edge of a particular tree to zero. In this case, the two nodes connected by this edge would collapse into one vertex with degree four. By expanding the resulting degree four vertex in an edge and two vertexes with degree three, a new tree topology can be created. This operation is also known as a nearest neighbor interchange (NNI) operation (Dinh et al., 2017; Steel, 2016). See figure 6.1 on page 121 for an example. By using this relation, orthants are called neighbors if there exists one NNI operation which can transform the one topology into the other. Neighboring orthants share a so called *face*. The face of an orthant in this case is a lower dimensional vector of branch lengths. The tree in the middle of figure 6.1 has a branch length vector with lower dimensionality than the left or right one.

Importantly, Billera et al. (2001) show, that this space of trees has certain properties such that a *geodesic* between two trees exists. Thus, the distance between two trees in

[5] Furthermore, there seems to be no intuitive interpretation of a tree with negative branch length in phylogenetic applications.

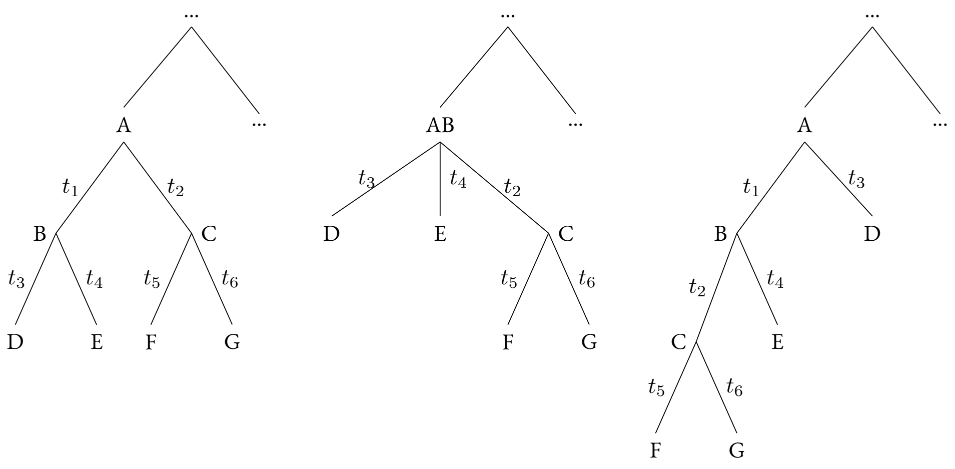


Figure 6.1: The rightmost tree can be generated from the leftmost by an NNI move. The tree in the middle displays the situation where the length of the branch t_1 is reduced to 0 from which the new tree can be generated.

this space is properly defined. Amenta et al. (2007) show how the lower and upper bound on this distance can be calculated in linear time and Owen (2008) presents an exact algorithm. Named after the authors of the original paper, this space of trees is called a Billera-Holmes-Vogtmann space.

This formulation of the space of trees can be used to define a Hamiltonian Monte Carlo sampler for (phylogenetic) trees. Since the space within one orthant is continuous, the traditional HMC procedure can be applied. It only becomes problematic when the boundary of an orthant is reached. Reaching the boundary of an orthant means that one particular element of the branch length vector q hits zero or becomes negative. Through the way the space of trees is defined, it is possible to interpret the scenario when an element of q becomes zero. In the case of a rooted binary tree, there are three topologies which share this particular face, two generated via an NNI operation and the original, which share this particular face. The sampler picks one of these three orthants at random and will continue the simulation in the newly selected orthant. The momentum corresponding to that particular attribute is then negated. Dinh et al. (2017) proof that the Hamiltonian remains unaffected by using the described technique. This defines the "leap-prog" integrator, the probabilistic counterpart to the standard leapfrog integrator. If multiple elements of q hit zero in a current update step, the random selection process is done for each element. Thus, it is possible to visit multiple tree topologies in just one leap-prog step.

As an extension to this simple algorithm, Dinh et al. (2017) note that via the discontinuity of the space of trees, the potential energy function on τ and q which is the likelihood function is not differentiable on the whole space and may thus lead to a loss of accuracy which cannot be neglected. To circumvent this problem, the authors propose to use a smooth approximation for (τ, q) such that each element of q is approximated such that the gradient vanishes when the element approaches 0. Dinh et al. (2017) propose the following smoothing function:

$$g_\delta(x) = \begin{cases} x, & x \geq \delta \\ \frac{1}{2\delta}(x^2 + \delta^2), & 0 \leq x \leq \delta. \end{cases} \tag{6.4}$$

This allows the derivative to be continuous across orthants. As a trade-off however, the likelihood function is not continuous anymore across orthants. To alleviate this problem, the state resulting from a topology move is accepted using the original Hamiltonian. As a side effect, a particular choice of δ also results in a lower bound for the individual branch lengths. The minimum of $g_\delta(x)$ is at $\frac{\delta}{2}$, which will effectively function as the lower bound. It can be shown that the Hamiltonian dynamics defined above uphold the reversibility, volume preservation and k-accessibility properties of traditional Hamiltonian dynamics (Dinh et al., 2017).

6.3 No-U-Turn Sampling in the Tree Space

Defining a No-U-Turn sampling scheme for (phyloegenetic) trees is straightforward given the theoretical groundwork laid out above. There are two aspects of NUTS which need to be adapted to define the pyhlogenetic No-U-Turn sampler (p-NUTS). The first part is to provide a HMC sampler in the space of trees and the second is to define an appropriate stopping rule for the NUTS tree building procedure.

Replacing the leapfrog integrator in the No-U-Turn sampler with the leap-prog integrator described in section 6.2 is straightforward. Since the leap-prog integrator preserves volume and is reversible like the leapfrog integrator, the replacement does not impact the target distribution. Dinh et al. (2017) prove the relevant properties which make the leap-prog integrator a valid HMC sampler. Thus, using the leap-prog integrator instead of the traditional leapfrog method maintains the theoretical properties of the original No-U-Turn sampler. Since Dinh et al. (2017) report a superior performance for the leap-prog integrator which uses the smoothed branch lengths, this variant will also be used for p-NUTS.

The second aspect which needs to defined is an appropriate stopping rule. For NUTS, the tree building procedure is stopped if an infinitesimal change in either direction would cause the states at the left- and rightmost part of the search tree to start moving closer together. For p-NUTS, this means a comparison of trees is necessary. Since the space of trees in which the algorithm operates has a distance measure to it, the geodesic is used for calculating these distances. Building on the definition of the lower and upper bound on the geodesic in the Billera-Holmes-Vogtman space by Amenta et al. (2007), these bounds are used as proxies for the actual distances. Using the lower and upper bounds as distance proxies results in the following procedure for checking the No-U-Turn behavior.

1. Simulate one leap-prog step for θ^- and θ^+ resulting in $\theta^{-'}$ and $\theta^{+'}$, respectively.
2. Calculate the lower bound on the distance of $\theta^{+'}$ and $\theta^{-'}$, D_l and the upper bound D_u for θ^+ and θ^-
    (a) if $D_l < D_u$ terminate the sampling step
    (b) else continue sampling.

Data Set	# sites	# languages
Dravidian	231	38
Hmong Mien	892	36
Timor Alor Panta	240	59

Table 6.1: Overview of the data sets used to test p-NUTS

Similar to standard NUTS, the simulation is also aborted if the error in the simulation becomes too large. The same rule as described above is used. This new stopping rule also leaves the target distribution intact since the target distribution is not affected by any of the operations in the No-U-Turn measure. The difference to standard NUTS lies in the more complex evaluation of $\theta^{-'}$ and $\theta^{+'}$. While the standard algorithm only requires the computation of two inner products, the p-NUTS sampler requires the evaluation of an entire leap-prog step. However, as soon as the height of the binary search tree j becomes larger than 2, computing two leap-prog steps is cheaper than computing another $2^{(j+1)}$ new leap-prog steps. This concludes the definition of the stopping rule for p-NUTS and thus the definition of the p-NUTS sampler. All the other steps of standard NUTS remain unchanged and thus the target distribution is not affected by the changes introduced here.

6.4 Empirical Evaluation

In order to asses the performance of the p-NUTS sampler, I conducted some empirical tests. I used three different datasets from the supplementary material of Jäger (2018) to infer posterior distributions over phylogenetic trees with the new p-NUTS method and MrBayes (Ronquist and Huelsenbeck, 2003), as the quasi "industry-standard". The assumption is that p-NUTS converges faster to the target distribution and yields a higher or similar estimated sample size for a smaller number of runs. Table 6.1 gives an overview over the relevant statistics of the data sets that were used to test p-NUTS. The data show different relations between the number of sites and the number languages. Since

	Dravidian					
	π	TL	LL	top.	gen.	top./gen.
MrBayes	0.781 ($\pm$ 0.014)	2.798 ($\pm$ 0.171)	-1604.046 ($\pm$ 7.18)	1×10^6	10^6	1
p-NUTS	0.778 ($\pm$ 0.014)	3.034 ($\pm$ 0.219)	-1620.548 ($\pm$ 11.16)	$\approx 2.58 \times 10^7$	5×10^5	≈ 51.6
	Timor Alor Pantar					
	π	TL	LL	top.	gen.	top./gen.
MrBayes	0.831 ($\pm$ 0.010)	3.535 ($\pm$ 0.20)	-2263.246 ($\pm$ 9.14)	1×10^6	10^6	1
p-NUTS	0.827 ($\pm$ 0.010)	3.754 ($\pm$ 0.250)	-2281.340 ($\pm$ 14.08)	$\approx 7.05 \times 10^6$	5×10^5	≈ 14.1
	Hmong Mien					
	π	TL	LL	top.	gen.	top./gen.
MrBayes	0.820 ($\pm$ 0.006)	4.205 ($\pm$ 0.109)	-8862.094 ($\pm$ 6.49)	1×10^6	10^6	1
p-NUTS	0.819 ($\pm$ 0.005)	4.257 ($\pm$ 0.124)	-8871.125 ($\pm$ 9.08)	$\approx 3.6 \times 10^6$	5×10^5	≈ 7.3

Table 6.2: Results from the MCMC runs for MrBayes and p-NUTS. π is the equilibrium frequency of the presence of the character, TL is the tree length, LL is the log-likelihood of the model, top. is the number of tree topologies visited by the chain, gen. is the number of generations the chain was run and top./gen. are the topologies visited per generation. For π, TL and LL, the mean values of the posterior distributions are reported here, the values in the brackets are the standard deviations.

the aim is not to infer a perfect phylogenetic model for the three different data sets but to test how well p-NUTS can sample tree structures, the only parameters inferred are the trees and the equilibrium frequency of the presence/absence of the binary coding character.

Table 6.2 provides a summary of the results of the different analyses.[6] The table clearly indicates that p-NUTS converges much faster on the desired posterior distributions than MrBayes. This is somewhat expected as already the original NUTS sampler is able to produce nearly independent samples in every generation. This behavior is mirrored by p-NUTS as can be seen by the fast convergence. Dinh et al. (2017) already showed that the probabilistic-path Hamiltonian Monte Carlo scheme is able to outperform MrBayes in this regard. The number of tree topologies visited also shows the improved sampling behavior of the p-NUTS in comparison to the standard Markov Chain Monte Carlo setting used by MrBayes. The MCMC sampler used by MrBayes in the current experiment uses four different moves to propose a new tree topology per

[6]It has to be kept in mind, that p-NUTS has a lower bound on the branch lengths through the usage of the smoothing parameter δ. This may lead to slightly different estimates in the exact branch lengths of the tree.

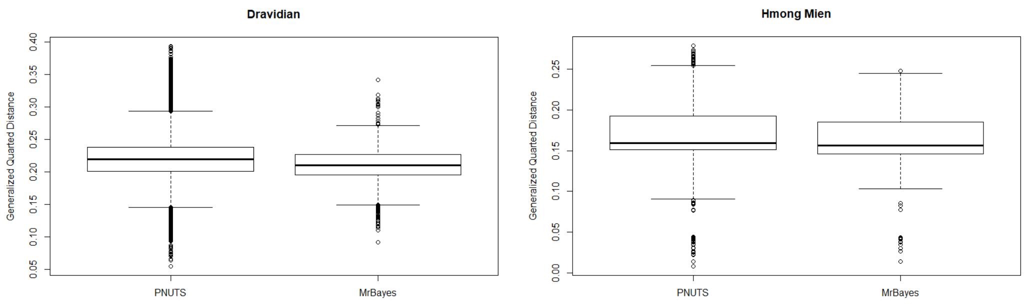


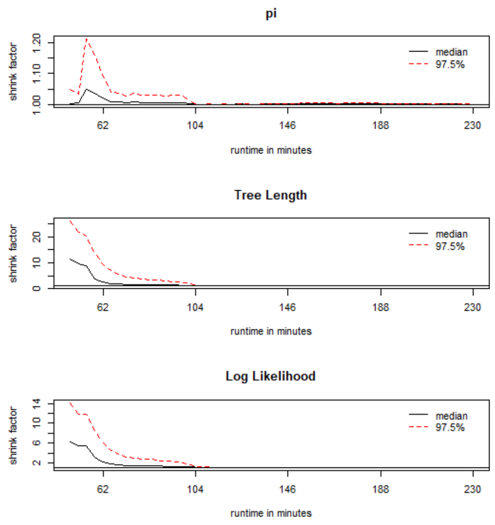
Figure 6.2: Distribution of GQD scores between the gold standard tree and the trees in the posterior sample of PNUTS and MrBayes.

generation. Overall, the data in the table shows that p-NUTS and MrBayes converge to similar posterior distributions. This statement is also confirmed by comparing the distributions of GQDs of the trees from the respective posterior samples to the gold standard trees (cf. figure 6.2).

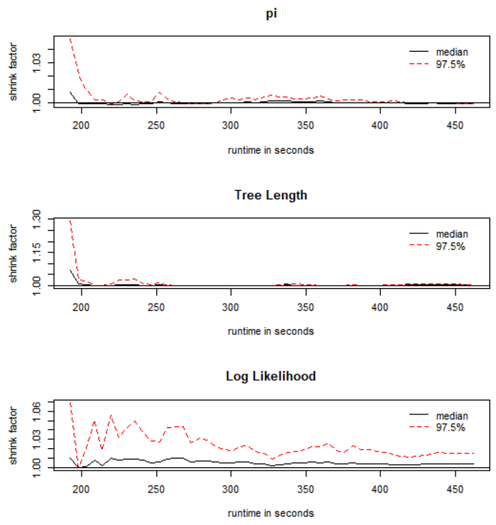
It has to be noted that p-NUTS is able to cycle through a considerably larger number of topologies per generation. Additionally, since MrBayes uses the Metropolis-Hastings sampler, only a part of the trees which are proposed are actually accepted. On the contrary, all the tree topologies which p-NUTS cycles through in its current implementation are accepted via the Hamiltonian.

6.4.1 Computational Performance

Although p-NUTS needs a considerably smaller amount of generations to sample from the target distribution, the current algorithm is not yet up to speed or even faster than MrBayes. Due to the design of the algorithm, the likelihood function needs to be evaluated for each proposed tree move. Consequently, for all the experiments, the number of tree topologies visited by p-NUTS is the minimal number of times the likelihood function is evaluated. As can be seen in table 6.2 on page 125, this number is always considerably larger than the number of generations in the experiments done with MrBayes. In addition to calculating the likelihood function, p-NUTS requires several eval-



(a) Shrink factor against time for p-NUTS



(b) Shrink factor against time for MrBayes

Figure 6.3: These graphs plot Gelman and Rubin's shrink factor against the elapsed run time of the MCMC chain for the experiment with the Dravidian data set using p-NUTS and MrBayes. (Gelman and Rubin, 1992; Brooks and Gelman, 1998)

uations of the gradient, which adds further computational burden. Implemented in the *Julia* programming language (Bezanson et al., 2017), the current version of p-NUTS can calculate $100,000$ generations in three hours and 50 minutes for the Dravidian data set, which amounts to roughly 435 generations per minute.[7] In addition to the costly p-NUTS sampler for the phylogenetic tree, the equilibrium frequencies are sampled using a slice sampler. The slice sampler may also require multiple evaluations of the likelihood function per generation. On the same machine, MrBayes can finish the calculations in a matter of minutes.[8] Figure 6.3 displays the evolution of Gelman and Rubin's shrink factor against run time (Gelman and Rubin, 1992; Brooks and Gelman, 1998). Even though MrBayes runs considerably faster and as a function of time the shrink factor

[7]Times are measured on an *AMD Ryzen 9 3900X CPU* with 12 cores and 2.9 GHz. Although the particular version of p-NUTS used only 4 cores maximally.

[8]On a side note, MrBayes is implemented in the C programming language. To compare p-NUTS and MrBayes on equal footing, both approaches would need to be written in the same programming language. It is, however, unlikely that this would change the quality of the runtime comparison because of the increased complexity of p-NUTS.

approaches convergence faster, p-NUTS also converges rapidly to a shrink factor of 1. In addition, the convergence of the log likelihood is much smoother in the case of p-NUTS.

As identified above, the two main bottlenecks for the current implementation of p-NUTS are the high number of evaluations of the likelihood function as well as the computation of the gradient. In its current implementation, p-NUTS uses an automatic differentiation library. It needs to be determined if a static version of the gradient can yield higher performance. Moreover, it is desirable to find a way to eliminate the frequent calculations of the likelihood function. Solving this issue may eventually also spill over to the stopping rule, which in its current state requires the simulation of a full leap-prog step. For further improvement of the p-NUTS algorithm, the second issue seems to be the most important one.

6.5 Conclusion

In this chapter I proposed an algorithm which is capable of performing No-U-Turn sampling for phylogenetic trees. Despite the complicated structure of the space of trees, Billera et al. (2001) were able to define the space of trees in such a way that Hamiltonian Monte Carlo sampling is possible. The leap-prog approach for sampling phylogenetic trees brought forward by Dinh et al. (2017) lends itself naturally to build an algorithm for phylogenetic No-U-Turn sampling. The results show that indeed better mixing and faster convergence is achieved by using p-NUTS to sample phylogenetic trees.

7 Conclusion

This thesis presented three different avenues by which the phylogenetic methods used in computational historical linguistics can be improved. On the one hand, there is the model presented in chapter 5, which aims to include explicit phylogenetic information into cognate detection. This can be seen as a step towards moving from the traditional two step approach of CHL to a one step approach as it is implicit in the traditional procedure in the comparative method. On the other hand, chapters 4 and 6 took a different way to adjust existing methods. While chapter 4 shows how an online training procedure is able to improve the training of existing cognate identification algorithms, it also argues that the current quality of automatically generated cognate clusters is already sufficient for phylogenetic inference. Especially the second part is important for the advancement of CHL since expert labeled cognate sets are hard to come by and are very labor intensive. Building on recent advancements in the area of Hamiltonian dynamics for non-continuous spaces, chapter 6 presents an algorithm which is capable of performing No-U-Turn sampling for phylogenetic trees. Hamiltonian dynamics are known to perform well for highly correlated parameters and it is thus important to make them available for phylogenetic trees.

In chapter 4, I showed that current cognate detection algorithms benefit considerably from optimized training regimes. Even in comparison to hand crafted alignment weights, the online training method showed superior performance in detecting cognate classes. This opens up the usage of automatic cognate identification tools for understudied languages. I showed that training on a large but unrelated dataset is sufficient to generate good cognate judgments. Especially, if these cognate sets are used in the downstream task of phylogenetic inference. Chapter 4.4 displayed an up to equal quality of phylogenetic trees from gold standard or inferred cognate data.

Building on the promising results of the preceding chapter, chapter 5 presented a hierarchical Bayesian model which includes phylogenetic information into a cognate detection task. The model introduces a time dependent notion of alignment which is build on an explicit model of sequence evolution. The parameters of this model are estimated using Markov Chain Monte Carlo techniques. This results in a posterior distribution over alignment scores. I used these scores to derive a distribution over cognate classes, which clearly distinguishes it from alternative approaches.

After introducing the slice-sampling approach to CHL for the time dependent alignment model, chapter 6 proposed a No-U-Turn sampler for phylogenetic tree structures. The standard tools for phylogenetic inference in CHL almost exclusively use random walk Markov Chain Monte Carlo methods. They are known to have slow mixing and convergence. Building on recent advancements which translates Hamiltonian Monte Carlo techniques such that they can be used for phylogenetic trees, I developed a prototype for a No-U-Turn sampler for phylogenetic trees. The algorithm indeed sampled from the correct distribution and reached faster convergence.

7.1 Outlook and Future Work

There are different ways to extend the work presented here. Obviously, since the time dependent alignment model did not fully live up to the expectations, some future work in this direction is necessary. As already discussed in the corresponding chapter, this model only approximates a posterior distribution over cognate classes. It is but a first step towards a full-scale Bayesian model of cognate class evolution. Another goal would be to introduce phylogenetic inference into the model to infer cognacy and phylogeny from the raw wordlist.

Another avenue to extend the results and methods presented in this thesis, is concerned with the p-NUTS algorithm. The No-U-Turn measure requires the explicit calculation of the next leap-prog step. This may result in a substantial computational overhead in the long run. It would be desirable to find a more refined way to calculate the No-U-Turn measure which has the same or similar predictive power as the current version. This may actually go hand in hand with eliminating several computations of the likelihood function which are done when walking through the space of trees. On the technical side, the partial derivative of the likelihood function becomes increasingly costly to compute for larger trees. This makes the algorithm in its current implementation unattractive for larger data sets. Thus, an efficient calculation of the partial derivative increases the usability of p-NUTS considerably.

7.2 Final Remarks

The main contribution to the field of computational historical linguistics surely are rather to methodology than to the results themselves. It has been shown that the toolbox of computational historical linguistics may still be constrained to a small set of instruments. However, these instruments can be refined in order to achieve better and sometimes faster results. It is, however, important to develop tools and models which maintain explainability and which can be tested explicitly. By developing such models and instruments, the reliability and quality of the results and theories can be strengthened.

References

Amenta, N., Godwin, M., Postarnakevich, N., and John, K. S. (2007). Approximating geodesic tree distance. *Information Processing Letters*, 103(2):61 – 65.

Amigó, E., Gonzalo, J., Artiles, J., and Verdejo, F. (2009). A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 12(4):461–486.

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*, 50(1/2):5–43.

Andrieu, C. and Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373.

Anthony, D. W. (2010). *Horse, the Wheel, and Language*. Princeton University Press.

Arunapuram, P., Edvardsson, I., Golden, M., Anderson, J. W. J., Novák, Á., Sükösd, Z., and Hein, J. (2013). StatAlign 2.0: combining statistical alignment with RNA secondary structure prediction. *Bioinformatics*, 29(5):654–655.

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

Baum, L. and Sell, G. (1968). Growth functions for transformations on manifolds. *Pac. J. Math.*, 27(2):211 – 227.

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In Shisha, O., editor, *Inequalities III: Proceedings of the Third Symposium on Inequalities*, pages 1–8, University of California, Los Angeles. Academic Press.

Baum, L. E. and Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73(3):360–363.

Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Statist.*, 37(6):1554–1563.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

Betancourt, M. (2013a). A general metric for riemannian manifold hamiltonian monte carlo. In *International Conference on Geometric Science of Information*, pages 327–334. Springer.

Betancourt, M. (2013b). Generalizing the no-u-turn sampler to riemannian manifolds. *arXiv stat.ME*.

Betancourt, M. (2018). A conceptual introduction to hamiltonian monte carlo. *arXiv stat.ME*.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):56–98.

Billera, L. J., Holmes, S. P., and Vogtmann, K. (2001). Geometry of the space of phylogenetic trees. *Advances in Applied Mathematics*, 27(4):733–767.

Borodovsky, M. and Ekisheva, S. (2006). *Problems and solutions in biological sequence analysis*. Cambridge University Press, 1 edition.

Bouchard-Côté, A., Hall, D., Griffiths, T. L., and Klein, D. (2013). Automated reconstruction of ancient languages using probabilistic models of sound change. *Proceedings of the National Academy of Sciences*, 10.1073/pnas.1204678110.

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

Brooks, S. P. and Gelman, A. (1998). General Methods for Monitoring Convergence of Iterative Simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455.

Brown, C. H., Holman, E. W., and Wichmann, S. (2013). Sound correspondences in the world's languages. *Language*, 89(1):4–29.

Brown, C. H., Holman, E. W., Wichmann, S., and 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.

Chang, W., Cathcart, C., Hall, D., and Garrett, A. (2015). Ancestry-constrained phylogenetic analysis supports the indo-european steppe hypothesis. *Language*, 91(1):194–244.

Chao, K.-M. and Zhang, L. (2008). *Sequence Comparison: Theory and Methods (Computational Biology)*. Springer.

Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Covington, M. A. (1996). An Algorithm to Align Words for Historical Comparison. *Computational Linguistics*, 22(4):481–496.

Cowles, M. K., Yan, J., and Smith, B. (2009). Reparameterized and Marginalized Posterior and Predictive Sampling for Complex Bayesian Geostatistical Models. *Journal of Computational and Graphical Statistics*, 18(2):262–282.

Croft, W. (2001). *Explaining Language Change*. Pearson Education ESL.

Damlen, P., Wakefield, J., and Walker, S. (1999). Gibbs sampling for bayesian non-conjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(2):331–344.

Darwin, C. (1871). *The descent of man and selection in relation to sex*. Murray, London.

Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C. (1978). A model of Evolutionary Change in Proteins. In *Atlas of protein sequence and structure 5*. Nat. Biomed. Res. Found.

Dellert, J. (2015). Compiling the Uralic Dataset for NorthEuraLex, a Lexicostatistical Database of Northern Eurasia. *Septentrio Conference Series*, 0(2):34–44.

Dellert, J. (2016). Using Causal Inference To Detect Directional Tendencies In Semantic Evolution. In Roberts, S., Cuskley, C., McCrohon, L., Barceló-Coblijn, L., Fehér, O., and Verhoef, T., editors, *The Evolution of Language: Proceedings of the 11th International Conference (EVOLANGX11)*. Online at `http://evolang.org/neworleans/papers/139.html`.

Dellert, J. (2017). *Information-Theoretic Causal Inference of Lexical Flow*. PhD thesis, University of Tuebingen.

Dellert, J. and Buch, A. (2018). A new approach to concept basicness and stability as a window to the robustness of concept list rankings. *Language Dynamics and Change*, 8(2):157–181.

Dellert, J. and Jäger, G. (2017). NorthEuraLex (version 0.9).

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Dinh, V., Bilge, A., Zhang, C., and Matsen IV, F. A. (2017). Probabilistic Path Hamiltonian Monte Carlo. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney.

Dolgopolsky, A. B. (1964). Gipoteza drevnejšego rodstva jazykovych semej Severnoj Evrazii s verojatnostej točky zrenija. *Voprosy Jazykoznanija*, 2:53–63.

Dolgopolsky, A. B. (1986). A probabilistic hypothesis concerning the oldest relationships among the language families of Northern Eurasia. In Shevoroshkin, V. V., editor, *Typology, Relationship and Time*, pages 27–50. Karoma Publisher, Ann Arbor.

Downey, S. S., Hallmark, B., Cox, M. P., Norquest, P., and Lansing, J. S. (2008). Computational feature-sensitive reconstruction of language relationships: Developing the aline distance for comparative historical linguistic reconstruction. *Journal of Quantitative Linguistics*, 15(4):340–369.

Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. (1987). Hybrid monte carlo. *Physics Letters B*, 195(2):216 – 222.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (2001). *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge Univ. Press, Cambridge, repr. edition.

Dyen, I., Kruskal, J. B., and Black, P. (1992). An Indo-European classification: A lexicostatistical experiment. *Transactions of the American Philosophical Society*, 82(5):1–132.

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

Felsenstein, J. (1973). Maximum Likelihood and Minimum-Steps Methods for Estimating Evolutionary Trees from Data on Discrete Characters. *Systematic Biology*, 22(3):240–249.

Felsenstein, J. (1981). Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of molecular evolution*, 17(6):368–376.

Felsenstein, J. (2004). *Inffering Phylogenies*. Sinauer Associates, Sunderland.

Felsenstein, J. and Churchill, G. A. (1996). A hidden markov model approach to variation among sites in rate of evolution. *Molecular Biology and Evolution*, 13(1):93–104.

Fitch, W. M. (2000). Homology. *Trends in Genetics*, 16(5):227–231.

Fitch, W. M. and Smith, T. F. (1983). Optimal sequence alignments. *Proceedings of the National Academy of Sciences*, 80(5):1382–1386.

Flouri, T., Kobert, K., Rognes, T., and Stamatakis, A. (2015). Are all global alignment algorithms and implementations correct? *bioRxiv*.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis*. Chapman and Hall/CRC.

Gelman, A. and Rubin, D. B. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457–472.

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705–708.

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

Greenhill, S. J. (2011). Levenshtein distances fail to identify language relationships accurately. *Computational Linguistics*, 37(4):689–698.

Greenhill, S. J., Blust, R., and Gray, R. D. (2008). The Austronesian Basic Vocabulary Database: from bioinformatics to lexomics. *Evolutionary bioinformatics online*, 4:271–283.

Greenhill, S. J., Drummond, A. J., and Gray, R. D. (2010). How accurate and robust are the phylogenetic estimates of Austronesian language relationships? *PloS one*, 5(3):e9573.

Greenhill, S. J. and Gray, R. D. (2009). Austronesian language phylogenies: Myths and misconceptions about Bayesian computational methods. *Austronesian Historical Linguistics and Culture History: A Festschrift for Robert Blust*, pages 375–397.

Haak, W., Lazaridis, I., Patterson, N., Rohland, N., Mallick, S., Llamas, B., Brandt, G., Nordenfelt, S., Harney, E., Stewardson, K., Fu, Q., Mittnik, A., Bánffy, E., Economou, C., Francken, M., Friederich, S., Pena, R. G., Hallgren, F., Khartanovich, V., Khokhlov, A., Kunst, M., Kuznetsov, P., Meller, H., Mochalov, O., Moiseyev, V., Nicklisch, N., Pichler, S. L., Risch, R., Guerra, M. A. R., Roth, C., Szécsényi-Nagy, A., Wahl, J., Meyer, M., Krause, J., Brown, D., Anthony, D., Cooper, A., Alt, K. W., and Reich, D. (2015). Massive migration from the steppe was a source for Indo-European languages in Europe. *Nature*, 522(7555):207–211.

Hammarström, H., Forkel, R., and Haspelmath, M. (2017). *Glottolog*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Hasegawa, M., Kishino, H., and aki Yano, T. (1985). Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolution*, 22(2):160–174.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Hauer, B. and Kondrak, G. (2011). Clustering semantically equivalent words into cognate sets in multilingual lists. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 865–873.

Hein, J. (2001). An algorithm for statistical alignment of sequences related by a binary tree. In *Biocomputing 2001*, pages 179–190. World Scientific.

Hein, J., Jensen, J. L., and Pedersen, C. N. (2003). Recursions for statistical multiple alignment. *Proceedings of the National Academy of Sciences*, 100(25):14960–14965.

Hein, J., Wiuf, C., Knudsen, B., Møller, M., and Wibling, G. (2000). Statistical alignment: computational properties, homology testing and goodness-of-fit1. *Journal of Molecular Biology*, 302(1):265–279.

Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, 89(22):10915–10919.

Higdon, D. M. (1998). Auxiliary Variable Methods for Markov Chain Monte Carlo with Applications. *Journal of the American Statistical Association*, 93(442):585–595.

Hoffman, M. D. and Gelman, A. (2014). The no-u-turn sampler: Adaptively setting path lenghts in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15:1593–1623.

Holman, E. W., Wichmann, S., Brown, C. H., Velupillai, V., Müller, A., and Bakker, D. (2008). Explorations in automated language classification. *Folia Linguistica*, 42(3-4).

Holmes, I. and Bruno, W. J. (2001). Evolutionary hmms: a bayesian approach to multiple alignment. *Bioinformatics*, 17(9):803–820.

Hruschka, D. J., Branford, S., Smith, E. D., Wilkins, J., Meade, A., Pagel, M., and Bhattacharya, T. (2015). Detecting Regular Sound Changes in Linguistics as Events of Concerted Evolution. *Current Biology*, 25(1):1–9.

Huelsenbeck, J. P. and Ronquist, F. (2001). MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics (Oxford, England)*, 17:754–755.

Huelsenbeck, J. P., Ronquist, F., Nielsen, R., and Bollback, J. P. (2001). Bayesian Inference of Phylogeny and Its Impact on Evolutionary Biology. *Science*, 294(5550):2310–2314.

Huff, P. and Lonsdale, D. (2011). Positing language relationships using aline. *Language Dynamics and Change*, 1(1):128–162.

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

Jäger, G. (2013). Phylogenetic Inference from Word Lists Using Weighted Alignment with Empirically Determined Weights. *Language Dynamics and Change*, 3(2):245–291.

Jäger, G. (2015). Support for linguistic macrofamilies from weighted sequence alignment. *Proceedings of the National Academy of Sciences*, 112(41):12752–12757.

Jäger, G. (2018). Computational Historical Linguistics. *CoRR*, abs/1805.08099.

Jäger, G. (2018). Global-scale phylogenetic linguistic inference from lexical resources. *Scientific Data*, 5:180–189.

Jäger, G. and List, J.-M. (2016). Statistical and computational elaborations of the classical comparative method. Manuscript, University of Tübingen.

Jäger, G., List, J.-M., and Sofroniev, P. (2017). Using support vector machines and state-of-the-art algorithms for phonetic alignment to identify cognates in multilingual wordlists. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1205–1216. Association for Computational Linguistics.

Jäger, G. and List, J.-M. (2018). Using ancestral state reconstruction methods for onomasiological reconstruction in multilingual word lists. *Language Dynamics and Change*, 8(1):22–54.

Jordan, M. I. (2004). Graphical models. *Statistical Science*, 19(1):140–155.

Jukes, T. H. and Cantor, C. R. (1969). Evolution of protein molecules. In Munro, H., editor, *Mammalian Protein Metabolism*, pages 21 – 132. Academic Press.

Karp, R. M. and Held, M. (1967). Finite-State Processes and Dynamic Programming. *SIAM Journal on Applied Mathematics*, 15(3):693–718.

Kimura, M. (1980). A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of molecular evolution*, 16(2):111–120.

Kimura, M. (1981). Estimation of evolutionary distances between homologous nucleotide sequences. *Proceedings of the National Academy of Sciences of the United States of America*, 78:454–458.

Kondrak, G. (2000). A new algorithm for the alignment of phonetic sequences. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, NAACL 2000, pages 288–295, Stroudsburg, PA, USA.

Kroonen, G. (2013). *Etymological Dictionary of Proto-Germanic*. Leiden Indo-European Etymological Dictionary Series. Brill Academic Publishers, Leiden.

Kruskal, J. B. (1983). An overview of sequence comparison. *SIAM Review*, 25(2):201–237.

Ladefoged, P. and Johnson, K. (2014). *A course in phonetics*. Nelson Education.

Lanave, C., Preparata, G., Saccone, C., and Serio, G. (1984). A new method for calculating evolutionary substitution rates. *Journal of molecular evolution*, 20:86–93.

Lees, R. B. (1953). The Basis of Glottochronology. *Language*, 29(2):113.

Lehmann, W. P. (1967). *A Reader in Nineteenth Century Historical Indo-European Linguistics*. Indiana University Press.

Leimkuhler, B. and Reich, S. (2005). *Simulating Hamiltonian Dynamics*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.

Levenshtein, V. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10(8):707–710.

Liang, P. and Klein, D. (2009). Online em for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 611–619, Stroudsburg, PA, USA. Association for Computational Linguistics.

List, J.-M. (2012a). LexStat: Automatic detection of cognates in multilingual wordlists. In *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*, pages 117–125. Association for Computational Linguistics.

List, J.-M. (2012b). Multiple sequence alignment in historical linguistics. In Boone, E., Linke, K., and Schulpen, M., editors, *Proceedings of ConSOLE XIX*, pages 241–260.

List, J.-M. (2012c). SCA. Phonetic alignment based on sound classes. In Slavkovik, M. and Lassiter, D., editors, *New directions in logic, language, and computation*, pages 32–51. Springer, Berlin and Heidelberg.

List, J.-M. (2014a). Investigating the impact of sample size on cognate detection. *Journal of Language Relationship*, 11:91–101.

List, J.-M. (2014b). *Sequence comparison in Historical Linguistics*. Düsseldorf University Press, Düsseldorf.

List, J.-M. (2015). Network perspectives on chinese dialect history. *Bulletin of Chinese Linguistics*, 8:42–67.

List, J.-M. and Forkel, R. (2016). Lingpy. a python library for historical linguistics.

List, J.-M., Greenhill, S. J., and Gray, R. D. (2017). The Potential of Automatic Word Comparison for Historical Linguistics. *PLOS ONE*, 12(1):e0170046.

List, J.-M., Lopez, P., and Bapteste, E. (2016). Using sequence similarity networks to identify partial cognates in multilingual wordlists. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 599–605, Berlin, Germany. Association for Computational Linguistics.

Longobardi, G., Guardiano, C., Silvestri, G., Boattini, A., and Ceolin, A. (2013). Toward a syntactic phylogeny of modern Indo-European languages. *Journal of Historical Linguistics*, 3(1):122–152.

Lunter, G., Drummond, A. J., Miklós, I., and Hein, J. (2005). Statistical alignment: Recent progress, new applications, and challenges. In *Statistical methods in molecular evolution*, pages 375–405. Springer.

Mackay, W. (2004). Word similarity using pair hidden markov models. master's thesis. Master's thesis, University of Alberta.

Mackay, W. and Kondrak, G. (2005). Computing word similarity and identifying cognates with pair hidden markov models. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 40–47, Stroudsburg, PA, USA. Association for Computational Linguistics.

McPherson, R. A. (2008). The numbers universe: An outline of the dirac/eddington numbers as scaling factors for fractal, black hole universes. *Electronic Journal of Theoretical Physics*, 5(18):81–94.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Metzler, D., Fleißner, R., Wakolbinger, A., and von Haeseler, A. (2001). Assessing variability by joint sampling of alignments and mutation rates. *Journal of molecular evolution*, 53(6):660–669.

Miklós, I. (2002). An improved algorithm for statistical alignment of sequences related by a star tree. *Bulletin of Mathematical Biology*, 64(4):771–779.

Miklós, I., Lunter, G., and Holmes, I. (2004). A "long indel" model for evolutionary sequence alignment. *Molecular Biology and Evolution*, 21(3):529–540.

Neal, R. M. (2003). Slice sampling. *The Annals of Statistics*, 31(3):705–767.

Neal, R. M. (2011). Mcmc using hamiltonian dynamics. In Brooks, S., Gelman, A., Jones, G., and Meng, X.-L., editors, *Handbook of Markov Chain Monte Carlo*. Chapman & Hall / CRC Press.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453.

Newman, M. E. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113.

Nichols, J. (1996). The Comparative Method as Heuristic. In Ross, M. and Durie, M., editors, *The Comparative Method Reviewed : Regularity and Irregularity in Language Change*, pages 39–71. Oxford University Press.

Novák, Á., Miklós, I., Lyngsø, R., and Hein, J. (2008). Statalign: an extendable software package for joint bayesian estimation of alignments and evolutionary trees. *Bioinformatics*, 24(20):2403–2404.

Osthoff, H. and Brugmann, K. (1878). *Morphologische Untersuchungen auf dem Gebiete der indogermanischen Sprachen*. Hirzel, Leipzig.

Owen, M. A. (2008). *Distance computation in the space of phylogenetic trees*. PhD thesis, Cornell University.

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

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rama, T. (2015). Automatic cognate identification with gap-weighted string subsequences. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*, pages 1227–1231.

Rama, T. (2018). Three tree priors and five datasets. *Language Dynamics and Change*, 8(2):182–218.

Rama, T. and Borin, L. (2015). Comparative evaluation of string similarity measures for automatic language classification. In Mačutek, J. and Mikros, G. K., editors, *Sequences in Language and Text*, pages 203–231. de Gruyter.

Rama, T., List, J.-M., Wahle, J., and Jäger, G. (2018). Are automatic methods for cognate detection good enough for phylogenetic reconstruction in historical linguistics? In *Proceedings of the North American Chapter of the Association of Computational Linguistics*, pages 393–400.

Rama, T., Wahle, J., Sofroniev, P., and Jäger, G. (2017). Fast and unsupervised methods for multilingual cognate clustering. *CoRR*, abs/1702.04938.

Redelings, B. D. and Suchard, M. A. (2005). Joint bayesian estimation of alignment and phylogeny. *Systematic Biology*, 54(3):401–418.

Renfrew, C. (1987). *Archaeology and language: The puzzle of Indo-European origins*. J. Cape.

Ringe, D., Warnow, T., and Taylor, A. (2002). Indo-european and computational cladistics. *Transactions of the Philological Society*, 100(1):59–129.

Ristad, E. and Yianilos, P. (1998). Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

Rodríguez, F., Oliver, J. L., Marín, A., and Medina, J. R. (1990). The General Stochastic Model of Nucleotide Substitution. *Journal of Theoretical Bology*, 142:485–501.

Ronquist, F. and Huelsenbeck, J. P. (2003). MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics (Oxford, England)*, 19:1572–1574.

Ronquist, F., van der Mark, P., and Huelsenbeck, J. P. (2009). Bayesian phylogenetic analysis using MRBAYES. In Lemey, P., Salemi, M., and Vandamme, A.-M., editors, *The Phylogenetic Handbook*, pages 210–266. Cambridge University Press.

Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420.

Ross, M. and Durie, M. (1996). Introduction. In Durie, M., editor, *The comparative method reviewed. Regularity and irregularity in language change*, pages 3–38. Oxford University Press, New York.

Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.

Schleicher, A. (1852). *Die Formenlere der kirchenslawischen Sprache*. König, Bonn.

Schleicher, A. (1861). *Compendium der vergleichenden Grammatik der indogermanischen Sprachen*. Böhlau, Weimar.

Schleicher, A. (1863). *Die Darwinsche Theorie und die Sprachwissenschaft*. Hermann Böhlau, Weimar.

Schleicher, A. (1983). *Die Sprachen Europas in systematischer Übersicht*. Number 4 in Amsterdam studies in the theory and history of linguistic science. John Benjamins Publishing Company, Amsterdam. (Reprint of the edition published in Bonn by König, 1850).

Sellers, P. (1974). On the Theory and Computation of Evolutionary Distances. *SIAM Journal on Applied Mathematics*, 26(4):787–793.

Smith, T. F., Waterman, M. S., and Fitch, W. M. (1981). Comparative biosequence metrics. *Journal of Molecular Evolution*, 18(1):38–46.

Steel, M. (2016). *Phylogeny*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Steel, M. and Hein, J. (2001). Applying the thorne-kishino-felsenstein model to sequence evolution on a star-shaped tree. *Applied Mathematics Letters*, 14(6):679 – 684.

Suchard, M. A., Lemey, P., Baele, G., Ayres, D. L., Drummond, A. J., and Rambaut, A. (2018). Bayesian phylogenetic and phylodynamic data integration using BEAST 1.10. *Virus Evolution*, 4(1).

Suchard, M. A. and Redelings, B. D. (2006). Bali-phy: simultaneous bayesian inference of alignment and phylogeny. *Bioinformatics*, 22(16):2047–2048.

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

Tamura, K. and Nei, M. (1993). Estimation of the number of nucleotide substitutions in the control region of mitochondrial dna in humans and chimpanzees. *Molecular biology and evolution*, 10(3):512–526.

Thorne, J. L., Kishino, H., and Felsenstein, J. (1991). An evolutionary model for maximum likelihood alignment of dna sequences. *Journal of Molecular Evolution*, 33(2):114–124.

Thorne, J. L., Kishino, H., and Felsenstein, J. (1992). Inching toward reality: An improved likelihood model of sequence evolution. *Journal of Molecular Evolution*, 34(1):3–16.

Torres, A., Cabada, A., and Nieto, J. J. (2003). An exact formula for the number of alignments between two DNA sequences. *DNA sequence : the journal of DNA sequencing and mapping*, 14(6):427–430.

Turchin, P., Peiros, I., and Gell-Mann, M. (2010). Analyzing genetic connections between languages by matching consonant classes. *Journal of Language Relationship*, 3:117–126.

Weiss, M. (2015). The comparative method. In Bowern, C. and Evans, N., editors, *The Routledge Handbook of Historical Linguistics*, pages 127–145. Routledge, New York.

Wichmann, S. and Holman, E. W. (2013). Languages with longer words have more lexical change. In *Approaches to Measuring Linguistic Differences*, pages 249–281. Mouton de Gruyter.

Wichmann, S., Holman, E. W., and Brown, C. H. (2018). The ASJP Database (version 18).

Wichmann, S., Holman, E. W., Rama, T., and Walker, R. S. (2011). Correlates of Reticulation in Linguistic Phylogenies. *Language Dynamics and Change*, 1(2):205–240.

Wieling, M. (2007). Comparison of dutch dialects. Master's thesis, University of Groningen.

Wieling, M., Leinonen, T., and Nerbonne, J. (2007). Inducing sound segment differences using pair hidden markov models. In Nerbonne, J., Ellison, M., and Kondrak, G., editors, *Proceedings of Ninth Meeting of the ACL Special Interest Group in Computational Morphology and Phonology*, Computing and Historical Phonology, pages 48–56, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wieling, M., Margaretha, E., and Nerbonne, J. (2012). Inducing a measure of phonetic similarity from pronunciation variation. *Journal of Phonetics*, 40(2):307–314.

Wieling, M., Prokić, J., and Nerbonne, J. (2009). Evaluating the pairwise string alignment of pronunciations. In *Proceedings of the EACL 2009 workshop on language technology and resources for cultural heritage, social sciences, humanities, and education*, pages 26–34. Association for Computational Linguistics.

Yang, Z. and Rannala, B. (1997). Bayesian phylogenetic inference using dna sequences: a markov chain monte carlo method. *Molecular Biology and Evolution*, 14(7):717–724.

Yanovich, I. (2017). Phylogenetic linguistic evidence and the Dené-Yeniseian homeland. Manuscript, University of Tübingen.

Zhang, C., Rannala, B., and Yang, Z. (2012). Robustness of Compound Dirichlet Priors for Bayesian Inference of Branch Lengths. *Systematic Biology*, 61(5):779–784.

Zharkikh, A. (1994). Estimation of evolutionary distances between nucleotide sequences. *Journal of molecular evolution*, 39:315–329.