Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Arts in Computational Linguistics

Automated Sound Law Inference Using Probabilistic Soft Logic

Author:
Thora Daneyko

July 2020

First Examiner:
Dr. Johannes Dellert

Second Examiner:
Prof. Dr. Gerhard Jäger

Eberhard Karls Universität Tübingen
Philosophische Fakultät
Seminar für Sprachwissenschaft

Antiplagiatserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst habe, dass ich keine anderen als die angegebenen Hilfsmittel und Quellen benutzt habe, dass ich alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe, dass die Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist, dass ich die Arbeit weder vollständig noch in wesentlichen Teilen bereits veröffentlicht habe, und dass das in Dateiform eingereichte Exemplar mit dem eingereichten gebundenen Exemplar übereinstimmt.

Tübingen, den _____ _____

Thora Daneyko

Abstract

For over a century, historical linguists have with great success reconstructed unattested ancestral languages and the sound changes that transformed them into our modern languages, using a procedure called the *comparative method*. Computational approaches to this task are almost non-existent to this date, especially for inferring *conditioned* sound change. In this thesis, I present SoInEn, a system for automated inference of conditioned sound laws from raw lexical data. It is implemented within *Probabilistic Soft Logic* (PSL), a framework for efficient inference on probabilistic graphical models specified as first-order logic rules. This should enable a rather direct translation of the human reasoning applied during the comparative method, and also allows SoInEn to comprehensibly explain its decisions.

While SoInEn is able to correctly identify several important sound laws for my test cases, Dravidian and Samoyedic, it turns out that even the ones with larger concept lists among the current lexical databases do not provide enough data for inferring more infrequent, highly context-dependent sound changes. Emulating complex human reasoning within PSL is also not as straightforward as anticipated due to the high interdependency of the different heuristics applied and the discrepancy between mathematical logic and human intuition.

These results highlight that for computationally solving sound law inference and ancestral language reconstruction, we need more lexical databases that focus on higher concept rather than language coverage. They also demonstrate PSL's potential as an alternative to "black box" machine learning approaches for solving complex inference tasks.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to Johannes Dellert, who not only supervised this thesis, providing me with invaluable advice, feedback and support, but has also influenced my academic development like no one else during the six years I have worked for him as a student assistant. It was he who sparked my interest in linguistic typology and historical linguistics, and gave me the opportunity to deepen my theoretical knowledge and expand my practical skills in (computational) linguistics during the always interesting work for him.

I also thank my fellow colleagues in the EtInEn group, Verena Blaschke, Zhuge Gao and Jekaterina Kaparina, for the interesting discussions, as well as Hizniye Isabella Boga, who also patiently listened to my complaints when something did not work (as was usually the case).

Finally, I would like to thank the many dedicated current and former lecturers at the Department of Linguistics, among them Christian Bentz, Armin Buch, Gerhard Jäger and Daniël de Kok, for showing me so many different perspectives on my subject, as well as Heike Oberlin from the Department of Indology for introducing me to Malayāḷaṁ.

Contents

Introduction

The idea that language change and diversification is governed to a large extent by regular sound changes (or *sound laws*) applying throughout the lexicon has fascinated historical linguists ever since its discovery in the second half of the 19th century. Even though the assumption that these sound laws are truly exceptionless has by now been falsified (Kiparsky 2003; Harrison 2003, e.g.), it is their quasi-regularity that has enabled us to reconstruct entire unattested ancestral languages via the *comparative method* (Crowley and Bowern 2010; Campbell 2013, e.g.). Following this iterative procedure, historical linguists deduce ancient word forms from the systematic similarities and differences between related modern word forms. Despite its age and the falsity of the regularity assumption, the comparative method is so far unsurpassed by any other manual or automated method for ancestral language reconstruction.

Early attempts at automating parts of the comparative method were still closely oriented towards the manual procedure. Due to its high complexity, more recent approaches have often shifted towards adopting more widely applicable methods from biology, graph theory or machine learning. While the earlier steps of the comparative method, such as cognacy judgment (identifying related inherited words) and loanword detection (identifying borrowed words) have received most attention from the computational linguistics community in the recent years, implementations of sound law inference in particularly are largely underrepresented. List (2019b) puts the task onto his list of "Open Problems in Computational Historical Linguistics", noting that it "has usually been overlooked greatly" (p. 14) and that "no current solutions exist" (p. 15). Even though the latter is not entirely true (see e.g. Hruschka et al. 2015), none of the proposed models can detect conditioned sound change, i.e. sound laws that only apply in specific contexts.

In this thesis, I present a system for automated (conditioned) sound law inference called SoInEn. It is designed as part of the larger EtInEn system, a suite of tools for various tasks in historical linguistics, such as morpheme splitting or loanword detection, that is currently being developed by a group at the General Linguistics department of the University of Tübingen. Like the other components of EtInEn, SoInEn is modeled inside the Probabilistic Soft Logic (PSL) framework. PSL converts first-order logic rules into an efficient probabilistic graphical model operating on soft truth values in the interval $[0, 1]$. This makes it possible to directly translate the reasoning applied by historical linguists within the comparative method into a probabilistic inference task.

In chapter 1, I give a general introduction to the methodological background surround-

ing sound change. I first present the *Neogrammarian Hypothesis*, the assumption that sound change is regular and exceptionless, and discuss its consequences and weaknesses. I then elaborate on the individual steps of the comparative method in detail, also treating its limitations. Afterwards, I shortly introduce a crucial prerequisite for computational applications of the comparative method, namely lexical databases, in particular the NorthEuraLex database which is used as a data source for SoInEn. Finally, I review previous implementations relating to the comparative method.

Chapter 2 then introduces PSL, in particular the syntax of its logical rules and their conversion into inference objectives. In the end, I briefly discuss why PSL is a good fit for computational historical linguistics and sound law inference in particular.

Due to its scarce treatment in computational linguistics, there does not yet exist a proper gold standard for sound law inference to evaluate performance against, especially not one pertaining to conditioned sound laws. In chapter 3, I therefore present the gold standard I have derived from the literature for the Dravidian language family and the Samoyedic branch of the Uralic language family, also discussing the individual challenges associated with each set.

In chapter 4, I then present SoInEn, elaborating its integration into the larger EtInEn system, the preprocessing steps that had be taken outside of PSL, and the logical predicates and rules formulated to represent the comparative method.

Afterwards, I evaluate SoInEn's performance on the Dravidian and Samoyedic data inside NorthEuraLex by comparing it to the gold standard derived earlier, and discuss the emerging issues in chapter 5.

Finally, chapter 6 concludes the thesis by outlining future work that has to be carried out in order to improve SoInEn as well as automated sound law inference in general. It also discusses the drawbacks and the potential of PSL for complex applications such as those in historical linguistics.

1 Methods in Phonological Comparative Linguistics

Comparative linguistics is a branch of *historical linguistics*. Historical linguists study language change, i.e. how the phonology, morphology, syntax and semantics of a language or group of languages has changed over time, reconstructing ancestral forms and deducing language relatedness (Campbell 2013). The most successful way of doing so has been the *comparative method*, which reconstructs earlier stages of a group of related languages by examining the differences and similarities between them (Campbell 2013; Crowley and Bowern 2010). Another method to investigate language change is *internal reconstruction*, which traces the evolution of a single language by looking at that language alone, e.g. reconstructing earlier forms of words from irregular word forms (Campbell 2013; Crowley and Bowern 2010).

Since the reasoning of SoInEn is based on the comparative method, I only discuss the comparison-based branch of historical linguistics in this thesis. I also disregard its application to morphology, syntax and semantics (i.e. anything except phonology), since SoInEn is designed to model phonological change only.

After introducing the format in which sound change rules are usually given (1.1), I discuss the idea of the *regularity of sound change* (1.2), which is a crucial precondition for the comparative method. Then, I explain the comparative method and how it is generally applied to investigate sound change (1.3.1). I also discuss its limitations and theoretical issues (1.3.2). Afterwards, I turn to lexical databases which are the foundation of more data-driven computational methods in historical linguistics (1.4). In particular, I introduce the NorthEuraLex database, whose data sets I have used in developing and evaluating SoInEn (1.4.1).

1.1 Expressing Sound Change

A certain sound change from mother language M to daughter language D is usually given in the format $s_M > s_D \ / \ E$, which means that the sound s_M of the mother language has become s_D in the daughter language in the environment or *context* of E in the mother

language (Campbell 2013; Crowley and Bowern 2010). In E, the underscore _ is usually used as a placeholder for the changing sound. k > t͡ʃ / _ i, for example, describes the palatalization of /k/ before the high front vowel /i/, a common sound change among the languages of the world. Expressions such as k > t͡ʃ / _ i are referred to as *sound laws*. The law-like properties of these sound changes are discussed in the next section.

Instead of specifying the sounds involved literally (k, t͡ʃ, i), feature sets or other shortcuts can be used. If the palatalization above applied before all front vowels, the sound law could be reformulated as k > t͡ʃ / _ [+front]. Intervocalic voicing of /t/ to /d/ could be expressed as t > d / V _ V. If it applied to all stop consonants, we could write [+stop,-voice] > [+voice] / V _ V.

All of the above sound laws are examples of *conditioned sound laws*, i.e. sound changes that only occurred in a limited environment. When a sound changed everywhere in a language regardless of phonetic context, we have an *unconditioned sound law*. Thus, l > r describes a sound law through which all /l/s of the mother language became /r/ in the daughter language.

The word boundary in contexts is marked by #. In a deletion or insertion, the missing sound is usually indicated by ∅. For example, ∅ > e / # _ CC describes the insertion of /e/ before word-initial consonant clusters. h > ∅ is the unconditioned deletion of all glottal fricatives.

1.2 The Regularity of Sound Change

Up until the 1800s, the study of language history and change mostly lacked systematic methods and consistent treatment, and was often driven by ideological intentions (Crowley and Bowern 2010). In the 1870s, a new school of linguistic scholars emerged in Germany, who called themselves the Neogrammarians (ger. *Junggrammatiker* 'young grammarians'). They posited what is now called the *Neogrammarian Hypothesis*, which states that 1) all sound changes are regular, i.e. apply without exception, and 2) are only conditioned by the phonetic environment and insensitive to the semantic or grammatical environment (Crowley and Bowern 2010; Hale 2007; Kiparsky 2003; Campbell 2013). This assumption enabled the systematic, falsifiable, and thus, scientific study of language change and laid the foundation for the comparative method (Crowley and Bowern 2010).

Originally, the idea that sound change operates without exceptions was meant literally, which is why the rules governing this change are referred to as sound laws (in analogy to natural laws such as in physics). This means that e.g. the sound law k > t͡ʃ / _ [+front] operates on all voiceless velar stops before front vowels, irrespective of e.g. word class, position in the sentence or meaning of the lexical items that contain it, and without any exceptions. The latter claim has been the main point of criticism towards the Neogrammarian Hypothesis ever since its introduction. In 1885, Schuchardt wrote in his paper with the telling subtitle "Against the Neogrammarians" (orig. "Gegen die Junggrammatiker"):

> "When a natural scientist hears of the exceptionlessness of the sound laws for the first time, he will probably think of sound laws that hold always and

> everywhere. [...] If you now inform this layman that such universal sound laws have not yet been discovered, that, in fact, all sound laws identified so far are characterized by a tight spatial and temporal constraint, he will miss that absolute necessity which always appears as a precondition for exceptionless laws."[1] (Schuchardt 1885, p. 9)

The spatial and temporal constraint that Schuchardt mentions refers to the fact that sound laws operate in different areas of the language community and in different times to a different extent. In every language, there usually exist several dialects whose boundary is not clear cut but blurred, which is why linguists usually speak of a *dialect continuum*. Each sound change takes place in a different subset of these dialects, and the areas in which individual sound changes apply overlap so much that it is often impossible to draw the line between two dialects with clearly distinct sets of sound laws (Schuchardt 1885; Crowley and Bowern 2010). It can even happen that along the lines of that dialect continuum, the sound change will gradually and seemingly arbitrarily affect more and more lexical items (Crowley and Bowern 2010). In view of this, it is hard to define multiple exceptionless sound laws operating in a language as a single fixed unit. This finally led the Neogrammarians to assume that sound laws operate on the level of individual speakers, which made Schuchardt wonder:

> "A sound change is often found within a very large area, i.e. in a number of connected dialects [...] Why should a sound change develop spontaneously in each of these speaker-specific languages that form a dialect?"[2] (Schuchardt 1885, p. 12)

It is not even the case that such a sound change develops spontaneously in any subdivision of a language and immediately applies to all parts of the lexicon. Often, it spreads from a single innovation in a few lexical items and gradually affects more and more parts of the lexicon until the sound change can be called complete. This process of words changing in analogy to other similar words is called *lexical diffusion* (Crowley and Bowern 2010; Kiparsky 2003). It shows that sound change, while at work, is neither exceptionless nor total, and if interrupted, the result will not be either (Kiparsky 2003).

In addition, the environment in which sound laws apply is not always purely phonetic in nature. There are changes that only occur at morpheme boundaries, in words of a specific word class (Crowley and Bowern 2010) or even just in individual lexical items without any apparent system, which is referred to as *sporadic change* (Campbell 2013).

Despite "[t]he falsity of the regularity assumption" being "evident" (Harrison 2003, p. 231), the Neogrammarian Hypothesis has given birth to the extremely successful comparative

[1] my translation, orig. "Wenn ein Naturforscher zum ersten Mal von der Ausnahmslosigkeit der Lautgesetze hört, so wird er wahrscheinlich an immer und überall geltende Lautgesetze denken. [...] Verständigt man nun jenen Laien darüber dass dergleichen allgemeine Lautgesetze noch nicht entdeckt sind, dass vielmehr allen bisher ermittelten Lautgesetzen eine verhältnissmässig enge räumliche und zeitliche Begrenztheit eignet, so wird er hier jene absolute Nothwendigkeit vermissen welche stets als Voraussetzung ausnahmsloser Gesetze erscheint."

[2] my translation, orig. "Ein Lautwandel findet sich oft über ein sehr weites Gebiet hin, d. h. in einer Reihe zusammenhängender Dialekte [...] Warum soll nun ein Lautwandel in jeder der Individualsprachen welche einen Dialekt ausmachen, spontan entstanden sein?"

method, the standard procedure for inferring sound change and reconstructing previous stages of a language (Campbell 2013; Crowley and Bowern 2010; Rankin 2003). In light of this, the regularity condition is usually interpreted more laxly in practice. Thus, in their introductory book to historical linguistics, Crowley and Bowern (2010) recommend to "[s]eparate those correspondences [sounds in the modern languages likely derived from a common ancestor] that are systematic from those that are isolated" (p. 168) and "simply ignore such isolated correspondences and reconstruct only on the basis of the evidence provided by systematic sound correspondences" (p. 169). Still, it is important to keep the theoretical weakness behind the comparative method in mind when applying it.

1.3 The Comparative Method

The Neogrammarian Hypothesis has a useful practical consequence: If a set of related languages is "generated" (at least on a phonological level) via successive application of several exceptionless sound laws to a common mother language, the regular output of these sound laws can be identified by comparison of the resulting daughter languages, and the original input of the mother language can be deduced from the characteristics of its reflexes in the daughter languages. This procedure is called the *comparative method* (Campbell 2013; Crowley and Bowern 2010; Rankin 2003).

Usually, the mother of a family or subfamily of languages is unrecorded. In such a case, it is called the *proto-language* of that family (e.g. "Proto-Germanic", "Proto-Uralic", "Proto-Indo-European") and may only hypothetically be reconstructed. Rarely, the proto-language or a close relative is known: Latin, for instance, is quite similar to what is reconstructed as Proto-Romance and can serve as a test case for the correctness of the comparative method (Campbell 2013). The unattested reconstructed words of the proto-language are conventionally prefixed with an asterisk * to mark them as hypothetical.

1.3.1 General Procedure

The comparative method is not an algorithm that has to be followed strictly. While there is large consensus in historical linguistics textbooks (Campbell 2013; Crowley and Bowern 2010; Rankin 2003) about the basic steps that should be taken when applying it, it is an individual process that also involves the linguist's intuition. After all, sound changes are not entirely regular, and the plausibility of reconstructed sound laws and proto-language inventories also depend on family-specific features such as areal effects.

In this section, I describe how a historical linguist would generally go about applying the comparative method, briefly discussing the challenges each step poses to automation. Initially, sets of related words that can be compared are identified. Afterwards, these words are aligned to discover which sounds of the daughter languages most likely go back to the same proto-sound. These reflexes can then be used to actually reconstruct the proto-sounds and establish a sensible proto-inventory. Finally, sound correspondences that are in complementary distribution or apparently evolving from the same proto-sound are investigated to detect the phonological context for conditioned sound laws.

tam	*maram*	maɾam	m	a	ɾ	a	m	nio	*bî'*	biːʔ	b	iː	ʔ
mal	*maraṁ*	maɾam	m	a	ɾ	a	m	yrk	*ju'*	juʔ	j	u	ʔ
kan	*mara*	mara	m	a	r	a	–	sel	*köt*	køt	k	ø	t

Figure 1.1: Left: Alignments for the Tamiḻ (tam), Malayāḷaṁ (mal) and Kannaḍa (kan) words for 'tree' (Burrow and Emeneau 1984); right: Alignments for the Nganasan (nio), Nenets (yrk) and Selkup (sel) words for 'ten' (Janhunen 1977).

1.3.1.1 Step 1: Establish Cognate Sets

The first step is to find sets of *cognates* among the languages being investigated, i.e. words that seem to be derived from the same ancestors (Campbell 2013; Crowley and Bowern 2010; Rankin 2003). Depending on the number and nature of the sound changes obscuring the relationship between the languages, this can be easy or rather difficult: It is obvious that Tamiḻ *maram*, Malayāḷaṁ *maram* and Kannaḍa *mara* (Dravidian), all meaning 'tree', are cognates (Burrow and Emeneau 1984). Nganasan *bî'*, Nenets *ju'* and Selkup *köt* (Samoyedic), all meaning 'ten', may seem completely unrelated at first sight, even though they have all developed rather regularly from Proto-Samoyedic **wüt* (Janhunen 1977).

This example already illustrates that the comparative method is an iterative process: To correctly identify all cognate sets, we need to know the sound changes that took place in all of the languages. However, these sound changes cannot be derived without comparing the members of cognate sets. In consequence, we will often come back to this step, finding new cognate sets or discarding previous ones, after we have established new sound laws in the later steps.

It is not always sufficient to look at words with the same meaning in all of the languages, as with the above examples. Meanings of words change over time, just like their appearance. English *walk* and German *walken* 'to knead' are both descendants of Proto-Germanic **walkan* 'to roll' despite their seemingly unrelated meanings (Kroonen 2013).

Another thing to look out for are loanwords. Consider Tamiḻ *pasu*, Malayāḷaṁ *paśu* and Kannaḍa *hasu*, all meaning 'cow'. These are clearly cognates and the word has partaken in the regular sound change p > h / # _ in Kannaḍa. However, neither *s* nor *ś* can be reconstructed as phonemes of Proto-Dravidian, as we will see in section 3.1. In fact, all three languages have borrowed this word from Sanskrit *paśu* 'cattle' (cf. Monier-Williams 1899). If we did not know that the three languages are completely unrelated to Indo-Aryan Sanskrit, but merely borrowed extensively from it, we could infer a wrong proto-phoneme and sound law for Proto-Dravidian and even falsely assume a genetic relationship between Dravidian and Indo-Aryan. Thus, after having applied the comparative method, we also need to revisit the cognate set to identify loanwords by looking for sounds and sound correspondences that are unexplainable given our current assumption about the phonological features of the proto-language.

deu	*Stein*	ʃtaɪ̯n	ʃ	t	aɪ̯	–	n	*Herz*	hɛɐ̯t͡s	h	ɛ	ɐ̯	t͡s	–
eng	*stone*	stəʊ̯n	s	t	əʊ̯	–	n	*heart*	hɑːt	h	ɑ	ɑ	t	–
isl	*steinn*	stɛi̯tn̥	s	t	ɛi̯	t	n̥	*hjarta*	çar̥tɑ	ç	ɑ	r̥	t	ɑ
nld	*steen*	steːn	s	t	eː	–	n	*hart*	ɦɑrt	ɦ	ɑ	r	t	–

Figure 1.2: Alignments for the German (deu), English (eng), Icelandic (isl) and Dutch (nld) words for 'stone' (left) and 'heart' (right) (Dellert and Jäger 2017).

1.3.1.2 Step 2: Find Sound Correspondences

After the cognate sets have been compiled, the words of each cognate set are *aligned* so that the sounds that probably correspond to each other in the different languages are all in one column (Campbell 2013; Crowley and Bowern 2010; Rankin 2003). Figure 1.1 shows such alignments for the Dravidian 'tree' and Samoyedic 'ten' examples from the previous section. The symbol – is used when a language has no sound that corresponds to the others in its column, either through deletion or insertion.

From these alignments we can directly derive the *sound correspondences* between our languages, i.e. the sets of sounds that have evolved from the same proto-sound (Campbell 2013; Crowley and Bowern 2010; Rankin 2003). In the example above, we get the correspondences m/m/m, a/a/a, ɾ/ɾ/r, a/a/a and m/m/– for the three Dravidian languages and b/j/k, iː/u/ø and ʔ/ʔ/t for the three Samoyedic languages.

Because human linguists have an intuitive understanding of which sounds are likely to correspond to each other, cognates are usually not explicitly aligned in manual applications of the comparative method. For computational approaches, however, alignments are indispensable, and automating this task is not trivial. The usual approach is to calculate substitution, deletion and insertion costs for every pair of symbols and align the words in such a way that the sum of these costs is minimized. However, since sounds can be more (like /t/ and /d/) or less similar (like /t/ and /o/) to each other, a fixed substitution cost is not feasible for phonetic alignment. Likewise, deletion and insertion is more likely for some sounds than for other. An alignment implementation should therefore ideally be informed about sound similarity and the likelihood of certain sound changes.

When the alignments and sound correspondences serve as input to another automatic process, there are also a few notational issues to be considered beforehand. The first is the question whether to treat long vowels and diphthongs as a single symbol or as several distinct sounds. In phonetic descriptions, they are usually viewed as single phonemes and often, this view is also reasonable for alignments. Consider the left alignment in Figure 1.2 of the Germanic words for 'stone': It is clear that the German, English and Icelandic diphthongs correspond to the Dutch long vowel and that there are no two correspondences a/ə/ɛ/e and ɪ̯/ʊ̯/i̯/e. The diphthong was already there in Proto-Germanic **staina* (Kroonen 2013) and it has remained a bimoraic vowel in its descendants, so there is no need to split them up.

Problems with this treatment arise when the long vowels or diphthongs have developed from other sounds in some of the daughter languages. Consider the right alignment in Figure 1.2: In German and English, the rhotic has been vocalized at the end of words and

tam	*cāṟṟu*	t͡ɕaːtːu	t͡ɕ	aː	tː	u	–	–	*kañci*	kaɲt͡ɕi	k	a	ɲ	t͡ɕ	i
mal	*cāṟṟuka*	t͡ɕaːtːuga	t͡ɕ	aː	tː	u	g	a	*kaññi*	kaɲːi	k	a	ɲ	ɲ	i
kan	*sāṟu*	saːru	s	aː	r	u	–	–	*gañji*	gaɲd͡ʑi	g	a	ɲ	d͡ʑ	i
tel	*cāṭu*	t͡ɕaːʈu	t͡ɕ	aː	ʈ	u	–	–	*ganji*	gãd͡ʑi	g	a	˜	d͡ʑ	i

Figure 1.3: Alignments for the Tamiḻ (tam), Malayāḷaṁ (mal), Kannaḍa (kan) and Telugu (tel) words for 'to publish, announce' (left) and 'rice gruel, starch' (right) (Burrow and Emeneau 1984).

fin	*hiiri*	hiːri	h	iː	r	i	nio	*śiti*	sʲiti	sʲ	i	t	i
est	*hiir*	hiːr	h	iː	r	–	enf	*side*	side	s	i	d	e
krl	*hiiri*	hiːri	h	iː	r	i	yrk	*cuɔʁ*	sʲidʲɑ	sʲ	i	dʲ	ɑ
vep	*hir'*	hirʲ	h	i	r	ʲ	sel	*šittį*	ɕitːɨ	ɕ	i	tː	ɨ

Figure 1.4: Left: Alignments for the Finnish (fin), Estonian (est), Karelian (krl) and Veps (vep) words for 'mouse' (Dellert and Jäger 2017); right: Alignments for the Nganasan (nio), Enets (enf), Nenets (yrk) and Selkup (sel) words for 'two' (Janhunen 1977).

before consonants, as in *Herz* and *heart*. This results in a diphthong in German and a long vowel in (British) English. Here, it makes sense to split the diphthong and long vowel and align their second parts to the rhotics in Icelandic and Dutch as ɐ̯/ɑ/ɐ̯/r.

The same question has to be answered for geminate consonants. Are these single phonemes or clusters of the same consonant? Treatment as a single unit is useful when a geminate was shortened in some daughter languages, as in the left Dravidian alignment in Figure 1.3: Here, Proto-Dravidian /tː/ became short /ɾ/ in Kannaḍa and short /ʈ/ in Telugu. It does not seem particularly fitting to align them as if a second short /t/ had been deleted in these two languages. On the other hand, geminates may often arise from consonant clusters, as in the right alignment: Here, the /t͡ɕ/ in the Proto-Dravidian */kaɲt͡ɕi/ was assimilated to /ɲ/ in Malayāḷaṁ, resulting in a sound correspondence t͡ɕ/ɲ/d͡ʑ/d͡ʑ that is clearly distinct from the preceding ɲ/ɲ/ɲ/˜.

Just as it often makes sense to separate long vowels or geminates, you might sometimes want to split a diacritic from its base symbol. In the right alignment in Figure 1.3, for instance, the nasalization of the first Telugu vowel has obviously developed from the deleted nasal. In this case, it is more adequate to place the nasalization diacritic in place of the following gap to receive an alignment ɲ/ɲ/ɲ/˜ instead of regarding õ as a single unit. The left alignment in Figure 1.4 of the Finnic words for 'mouse' is another example where it makes sense to treat the secondary articulation as a separate symbol: In Estonian and Veps, the final /i/ has been deleted, but in Veps, it has left a trace in that the final consonant has been palatalized. It seems fitting to therefore match the palatalization diacritic with the final vowels in Finnish and Karelian.

In cases where the "triggering" phoneme is not deleted, however, it is more useful to group the palatalization diacritic with its consonant, as in the right alignment in Figure 1.4 of the Samoyedic words for 'two'. In Nganasan and Nenets, the front vowel *i* and, only in Nenets, the front vowel *ä* in Proto-Samoyedic **kitä* (Janhunen 1977) have triggered palatalization of the preceding consonants, but were not deleted in the process. It makes more sense

here to view the palatalization as a feature of the consonant instead of having two columns with inserted palatalization diacritics.

When applying the comparative method manually, one can split these problematic combinations in one case and treat them as one in another, whichever makes more sense in each case. When generating or processing these alignments automatically, however, we have to settle on one treatment and apply it consistently. If we do not have the restricted use case of a single language family, this will necessarily lead to unsatisfying solutions for some alignments, and even within the same language family, conflicting treatments may be demanded by different cognate sets, as underlined by the Germanic examples in Figure 1.2.

1.3.1.3 Step 3: Reconstruct Proto Sounds

Using the sound correspondences found in the previous step, we can now reconstruct the phoneme inventory of the proto-language (Campbell 2013; Crowley and Bowern 2010; Rankin 2003). Under the regularity assumption, each sound correspondence is the output of an exceptionless sound law that has applied to a proto-phoneme, and the nature of this proto-phoneme can be inferred from the characteristics of its descendant sounds. Since we know that sound laws are not always entirely regular and that our data might be corrupted by false cognacy judgments, missing loanword annotations or bad alignments, we will discard rare and "strange" sound correspondences until we can satisfyingly explain them, and regard the more frequent correspondences first (Campbell 2013).

When all or the vast majority of sounds in a correspondence are the same, reconstructing the proto-phoneme is straightforward (Campbell 2013; Crowley and Bowern 2010; Rankin 2003). It is clear that the ancestor of the correspondence m/m/m observed in the Dravidian words for 'tree' in Figure 1.1 is */m/. Similarly, the proto-phonemes for initial t͡ɕ/t͡ɕ/s/t͡ɕ and p/p/h/p in the Dravidian correspondences in Figure 1.3 are likely /t͡ɕ/ and /p/.

The "majority vote" is not always correct, though. There are sound changes that occur frequently among the languages of the world (Campbell 2013; Crowley and Bowern 2010), such as stops turning into fricatives at the same (or a similar) place of articulation (*spirantization*), as in the Kannaḍa changes p > h and t͡ɕ > s just mentioned. Then there are sound changes that are almost never observed, such as the reverse of spirantization: Stops rarely develop out of fricatives, so if we had a correspondence h/h/p/h instead, we would still be inclined to reconstruct a /p/ as the proto-sound, because the change h > p is just too unlikely. For the same reasons, the most likely proto-candidate for the Samoyedic correspondence ʔ/ʔ/t in Figure 1.1 is /t/ and not /ʔ/: It is quite common for stops to be reduced to a glottal stop, but a glottal stop does usually not change its place of articulation.

In other cases, the proto-sound might not have been preserved in any of its descendants. While we should normally refrain from reconstructing phonemes unattested in our correspondences (Crowley and Bowern 2010), the sounds that we can observe in the daughter language are sometimes all unsatisfying proto-candidates. The Samoyedic correspondences b/j/k and iː/u/ø from the word for 'ten' in Figure 1.1 are such a case: If we assume any of the involved sounds as the proto-sound, we always get at least one improbable sound

change. To resolve this, we compare the features of the observed sounds: Of the three vowels, two are round, two are high and two are front. This suggests /y/ as the proto-candidate. Among the consonants, we observe a labial, a velar and an approximant sound. A phoneme that combines these three features would be the labio-velar approximant /w/. While the sound changes w > b and w > j / _ [+front] are plausible, w > k is still unusual. If we look at what happens to the palatal approximant /j/ in Selkup, however, we find evidence that it also becomes a voiceless obstruent, as in Nganasan *je*, Nenets *e*, Selkup *čö* 'jaw' (Janhunen 1977). Since sound changes often apply to several sounds of the same type analogically, this observation increases the likelihood of Selkup w > k, making /w/ a plausible proto-sound for b/j/k. Indeed, the Proto-Samoyedic word for 'ten' is reconstructed as **wüt* in the literature (Janhunen 1977).

This example shows that it is not sufficient to only regard the sound correspondence in question. The direct and wider context of a correspondence can give important hints as to which kind of sound is the most likely proto-reconstruction: Neighboring sounds, patterns across the entire word (such as *vowel harmony*) and analogical sound correspondences and changes should be consulted to arrive at a satisfying solution. Even more generally, newly assumed proto-sounds should keep the entire reconstructed proto-phoneme inventory plausible: Languages generally prefer balanced or "symmetric" vowel and consonant inventories, and there seem to be universal tendencies to include certain sound types as well as dependencies between the occurrences of some groups of sounds (Campbell 2013; Crowley and Bowern 2010). While these restrictions are usually not absolute, proto-inventories should not violate them without good reason.

1.3.1.4 Step 4: Detect Environments for Conditioned Sound Change

The sound correspondences together with their proto-reconstructions directly yield the first simple sound laws (Campbell 2013; Crowley and Bowern 2010; Rankin 2003). From the Finnic correspondence iː/iː/iː/i (cf. Figure 1.4) we can infer the unconditioned sound change iː > i for (South and Central) Veps and, with the evidence from other vowel correspondences, the more general [+long] > [-long] (Tuisk 2010). More often, however, the underlying sound changes are conditioned. The usual signal for this is that two or more sound correspondences are in *complementary distribution*, i.e. occur in mutually exclusive contexts (Campbell 2013; Crowley and Bowern 2010; Rankin 2003).

Usually, a sound law has only affected some of the languages compared. In such cases, the complementary sound correspondences are very similar, differing only in a few sounds. In Figure 1.2, for example, we have the sound correspondence t/t/t/t in the left alignment, but t͡s/t/t/t in the right alignment for the four Germanic languages. While it is possible that the second correspondence has descended from Proto-Germanic */t͡s/, we have hopefully considered */t/ as the more likely proto-candidate. This is a problem: Under the regularity assumption, */t/ cannot have remained /t/ in one German word but randomly changed to /t͡s/ in another. This must be a case of conditioned sound change.

When investigating the environments of the correspondences t/t/t/t and t͡s/t/t/t, we find that the mutation of single */t/ occurs word-initially (as in *Zahn* [t͡saːn] 'tooth' < **tanþ*), after a liquid, i.e. /l/ or /r/ (as in *Herz* above), and when it is geminate (as in *Schatz*

'treasure' < **skatta-*; Kroonen 2013). In the remaining cases, such as word-medially or after different consonants (as in *Stein*), we find German /t/. Thus, we can postulate the sound laws t > t͡s / # _ , t > t͡s / [+liquid] _ and tː > t͡s for German (the implicit default case being t > t for all remaining contexts).

Sometimes, however, a conditioned sound law has affected all or almost all daughter languages, obscuring the original proto-sound. The conditioned sound change might not be obvious in such a case, because the complementary sound correspondences might have nothing in common. The Samoyedic correspondence sʲ/s/sʲ/ɕ in Figure 1.4 is an example of this: It can seem straightforward to reconstruct */s/ or */sʲ/ as a proto-sound. When examining the environments in which this sound correspondence is found, though, we find that its occurrence is highly restricted to the position before */i/ and */e/. It is unlikely that */s/ or */sʲ/ exclusively occurred before these sounds in Proto-Samoyedic, so this might be a case of conditioned sound change. Searching for complementary sound correspondences, we end up with the following set[3] (Janhunen 1977):

- sʲ/s/sʲ/ɕ before */i/, */e/
- k/s/sʲ/ɕ before */y/, */ø/
- k/s/sʲ/q before */æ/
- k/k/χ/k before */u/
- k/k/χ/q before */ɑ/, */o/

It seems that the proto-sound for all these correspondences is actually */k/ and that palatalization took place in all four Samoyedic languages to varying degrees: Nganasan palatalized */k/ only before */i/ and */e/, Selkup before all non-low front vowels and Enets and Nenets before all front vowels. In addition, /k/ became /χ/ before back vowels in Nenets and /q/ before non-high vowels in Selkup. Overall, the correspondences above yield six conditioned sound laws to deal with Proto-Samoyedic */k/ in its four daughter languages.

While such a sound change is still rather easy to resolve for a human linguist despite its complexity, especially since the occurrence of a sibilant before front vowels should always be a warning flag for anyone trained in historical linguistics (the palatalization of /k/ to a sibilant is an extremely frequent sound change), a computer could have its difficulties with the above example. Since there are so many complementary correspondences that occur in such restricted contexts, the evidence in the data is sparse. In Janhunen's (1977) Samoyedic etymological dictionary, for example, there is just a single entry for word-initial */kø/ and five for */ky/, of which only two list reflexes for all four languages. This might not be enough for a system to recognize the correspondence k/s/sʲ/ɕ as regular instead of sorting it out as noise stemming from false cognacy judgments or wrong alignments. We should also not let our program try to unify arbitrarily many complementary correspondences to keep it from explaining a set of rare nonsensical correspondences as conditioned sound change.

[3]Restricted to word-initial correspondences to keep the example "simple": Other sound laws applying intervocally result in a different correspondence set for word-medial environments.

1.3.1.5 Step 5: Verify Your Theory

The comparative method is an iterative process (Campbell 2013; Rankin 2003), and after completing step 4, one will usually apply it again based on the newly gained insights. The discovered sound laws can lead to different cognacy judgments: A word that was previously considered inherited may now be classified as a loanword when it violates common sound changes. Alignments can be readjusted based on the regular sound correspondences that were identified. Proto sound reconstructions might have to be reconsidered when they violate the symmetry of the proto-language's phoneme inventory, while gaps in the inventory can give hints for sound correspondences of previously doubtful origin. Therefore, the comparative method should be applied over and over, until the descendant languages' data can be satisfyingly explained with respect to the developed proto-language (which may never be the case).

1.3.2 Shortcomings

The crucial premise of the comparative method is the regularity of sound change, the problem with which I have already discussed in section 1.2: Sound change is not, in fact, entirely regular or without exception; it can operate in different dialects of a language to different extents, may affect only parts of the lexicon, or be conditioned on non-phonetic factors. Therefore, there will always be phonetic forms in modern languages that cannot be explained by way of the comparative method.

To understand more of these residual phenomena, one often has to turn towards social or cultural factors instead of purely linguistic ones. Language contact can not only alter the lexicon via word borrowing: A group of languages from several unrelated families may form a *Sprachbund* (linguistic area), sharing phonological and grammatical characteristics that can give the impression of relatedness (Aikhenvald and Dixon 2001; Harrison 2003; Campbell 2013). Phonemes may be borrowed and even incorporated into native material for a variety of reasons: The Bantu languages of Southern Africa, for example, presumably borrowed click consonants from the surrounding Khoisan languages to adhere to linguistic taboo or mark their identity (Daneyko and Bentz 2019). Finally, the phonological features of surrounding prestigious languages can influence the outcome of sound change, making the introduction of sounds prominent in these languages more likely than they would normally be, as seems to be the case e.g. for the overly frequent development of tones in Mainland South-East Asia and Sub-Saharan Africa (Aikhenvald and Dixon 2001).

The most definite limit of the comparative method is the time depth to which it can be applied (Aikhenvald and Dixon 2001; Harrison 2003). Since languages change constantly, cognacy gets harder to detect the earlier the languages diverged, and with that, the comparative method becomes more difficult to apply. The time depth up to which we can reliably reconstruct proto-languages has often been cited to be about $8,000$ years back, though this depends strongly on the rate of sound change which is higher for some languages than for others (Aikhenvald and Dixon 2001; Harrison 2003). It is clear that an $8,000$ years old proto-language reconstructed using the comparative method is merely an artificial construct that will more or less closely resemble the language as it was spoken at that time (provided that it even existed).

Despite its limitations, "the comparative method is arguably the most stable and successful of all linguistic methodologies" (Rankin 2003, p. 208), still being widely applied over a century after its introduction. Even though sound changes are not regular per se, they become quasi-regular after enough time has passed for them to be applied throughout the whole lexicon, which is often the case (Harrison 2003). This provides enough of a basis for the comparative method to work quite reliably.

1.4 Lexical Databases

The only prerequisite for the comparative method is the availability of lexical material for the daughter languages. This need and the increasing application of computational methods in historical linguistics has given rise to a number of lexical databases providing translations of a predefined list of concepts for a large number of languages in a unified format. The concepts selected for these lists are usually intended to be basic, in that they have equivalents in most languages, and stable, in that they are rarely subject to change or borrowing. The number of concepts varies greatly between databases, from 40 in the wide-coverage *Automated Similarity Judgment Program* database (ASJP; Wichmann, Holman, and Brown 2020), $\sim$ 210 in the *Austronesian Basic Vocabulary Database* (ABVD; Greenhill, Blust, and Gray 2008), 225 in the *Indo-European Lexical Cognacy Database* (IELex; Dunn 2015), 607 in the Indonesian-based *LexiRumah* (Kaiping and Klamer 2019), to $\sim 1,000 - 2,000$ in the *World Loanword Database* (WOLD; Haspelmath and Tadmor 2009). While 100 – 200 lexical items per language indeed seem to be optimal for inferring language relatedness and phylogenies (Rama and Wichmann 2018), automatically applying the comparative methods to yield a complete set of sound laws probably requires more data.

1.4.1 NorthEuraLex

The source material for SoInEn is provided by NorthEuraLex, a recently developed database focusing on the languages spoken in Northern Eurasia (Dellert and Jäger 2017; Dellert, Daneyko, et al. 2020). Each of the 107 languages in the current version (0.9) covers a list of 1,016 concepts, which should constitute a good basis for detecting sound correspondences and deriving sound laws. Since the goal of NorthEuraLex is to also provide data for exploring language contact, the concept list it builds on is not restricted to basic vocabulary, but also contains concepts commonly borrowed, such as month names.

NorthEuraLex provides IPA transcription for all languages, which is generated automatically on the basis of handcrafted rules derived from phonological descriptions. These are supposed to produce the actual pronunciation of the language, not a phonemic representation, also modeling language-internal phonological processes such as assimilation or final devoicing. The phonetic transcription of the resulting data is thus very detailed, exceeding the basic phonemic representations usually used as input to the comparative method. It will be interesting to see how an automated system handles this.

In its current version, NorthEuraLex provides no cognate set or loanword annotations for its data, leaving this step of the comparative method to be solved computationally as

well. Resolving this issue will be particularly important considering the potentially higher density of loanwords compared to other lexical databases.

1.5 Automating the Comparative Method

Most applications in computational linguistics that relate to parts of the comparative method are centered around cognacy detection. These implementations usually proceed in two steps – measuring word pair similarity and clustering the most similar words into cognate sets – and differ primarily in the methods used to perform these two separate tasks. One of the most prominent linguistically motivated implementations is the LexStat algorithm of List (2012), which builds on Turchin, Peiros, and Gell-Mann's (2010) approach to convert the original word forms using a heavily reduced set of consonant-only symbols (*consonant-class matching*/CCM) before computing similarity scores. LexStat and an extension for detecting partial cognacy (List, Lopez, and Bapteste 2016) are included in the widely-used Python package LingPy which unifies several tools for computational historical linguistics (List and Moran 2013). Rama and List (2019) build on the CCM approach as well, but relax the strict matching condition by using skip-grams. Most of the other recent cognate detection systems are based on machine learning, however (e.g. Hall and Klein (2010), Rama (2016), Jäger, List, and Sofroniev (2017), Rama, Wahle, et al. (2017), Hämäläinen and Rueter (2019), and Labat and Lefever (2019)).

The related task of loanword detection has generally received less treatment. Many implementations focus on detecting borrowing between specific unrelated languages or language families (e.g. Mi, Yang, Zhou, et al. (2016), Mi, Yang, Wang, et al. (2018a), and Mi, Yang, Wang, et al. (2018b) for Chinese, Russian and Arabic loans in Uyghur; or Mennecier et al. (2016) for borrowings between Turkic and Indo-Iranian languages), usually building on the assumption that words of similar form and meaning shared by unrelated languages must be loanwords. The LingPy implementation is language-independent, but also attributes loanword status to detected "cognates" between different branches of a given language tree only (List and Moran 2013). Similarly, Köllner and Dellert (2016) consider a word for a concept borrowed when the direct ancestor's translation of that concept is in a different cognate set, which does not capture all possible types of intra-family borrowings either. Minett and Wang (2003) directly aim at detecting loanwords between related languages, but proceed similarly to the LingPy implementation, with the difference that they infer the family tree themselves in the process.

Much less work in computational historical linguistics has been done on proto-form reconstruction and sound law inference. One of the earliest attempts to reproduce the comparative method computationally was by Lowe and Mazaudon (1994), who implemented the "Reconstruction Engine", a program that could infer cognate sets and proto-forms from modern lexical data, but which also required the user to provide the regular sound correspondences with their proto-phonemes. A more holistic approach was chosen later by Oakes (2000), who employed dynamic programming algorithms to perform cognacy judgments, alignment, sound correspondence detection, proto-phoneme reconstruction and, finally, proto-word reconstruction for Proto-Malayo-Javanic. While his method was able

to detect when sound changes were conditioned, he did not derive the contexts in which they occurred.

More recent attempts to automate (parts of) the comparative method do not stick so closely to the manual procedure: List (2019a) infers regular sound correspondences by reducing the task to the minimum clique cover problem from graph theory, but did not take the next step of reconstructing proto-phonemes. Bouchard-Côté et al. (2013) employ probabilistic model of sound change combined with a Monte Carlo inference algorithm for (optionally) cognate detection and proto-form reconstruction, also producing sound changes for individual cognate sets along the way, though these are not the primary target. Finally, Hruschka et al. (2015) set out to infer unconditioned sound laws for the Turkic languages using a Markov Chain Monte Carlo model, adopting ideas from biology pertaining to the concept of concerted evolution.

Overall, a system that models the comparative method in its entirety up to the production of sound laws has yet to be implemented. Recent approaches to proto-reconstruction and sound law inference have diverged from the original method, instead leveraging algorithms from biology or machine learning. In particular, the automated detection of conditioned sound laws is currently completely unsolved to my knowledge.

2 Probabilistic Soft Logic

Probabilistic Soft Logic (PSL) is a reasoning framework over probabilistic graphical models that are specified using first-order logic rules with soft truth values in the interval $[0, 1]$. These rules can be translated into hinge-loss Markov random fields (HL-MRFs), for which most probable variable assignments can be found using fast, accurate and highly scalable optimization algorithms (Bach et al. 2017).

The PSL implementation used by the EtInEn group and for the model presented in this thesis was developed by the LINQS group (LINQS Research Group 2018) and is available as an open source Java library which provides fast, efficient algorithms for inference and rule weight learning, supporting multi-threading and different SQL database backends. It is still in active development with new improved releases coming out regularly. For my thesis project, I am using version 2.1.0 of LINQS PSL.

In section 2.1, I introduce the syntax of the PSL modeling language in which the predicates and rules of the PSL models are specified. Some rule engineering techniques specific to the LINQS implementation and their grounding process are discussed in section 2.2. I will then explain how these first-order logic rules are translated into HL-MRFs for efficient inference in section 2.3. Finally, in section 2.4, I discuss why PSL is a promising framework for historical linguistics.

2.1 PSL Syntax

This section introduces the PSL modeling language. It mainly revolves around two concepts: The individual facts, represented by predicates or, more specifically, atoms, and the rules that relate these facts to each other.

2.1.1 Predicates and Atoms

A PSL *predicate* is a relation over a fixed number of terms (as specified by its *arity*). `PartOfSpeech`, for example, could be a predicate with arity 4 for a part-of-speech

tagger, the arguments being a sentence id, the index of a word in that sentence, the word itself, and its part of speech. In rules, this predicate can be referred to as e.g. `PartOfSpeech(SentenceID, Idx, Word, PoS)`, where `SentenceID`, `Idx`, `Word` and `PoS` are *variables*, placeholders for actual values to be inserted. The arity can be any positive integer.

A (ground) *atom* then is an actual instance of a predicate where *constant* values have been inserted for the variables. Examples of atoms for `PartOfSpeech` are `PartOfSpeech("1", "3", "tree", "noun")` or `PartOfSpeech("13", "5", "small", "adjective")`. In LINQS syntax, constants are given in quotation marks to distinguish them from variables. An atom is associated with a *belief value* in the range $[0, 1]$ that specifies the current truth value of the relation encoded by the atom. Belief values are either specified beforehand or inferred by the system.

Atoms with a fixed belief value are called *observations*. They are the input to the PSL model: Known facts that can be used to draw conclusions over the unknown data, the *target* atoms, whose belief values are inferred by PSL. Both observations and targets must be specified by the user, since PSL cannot be expected to come up with plausible ideas for targets on its own. In our PSL part-of-speech tagger, we could, for instance, enter `PartOfSpeech("1", "3", "tree", "noun")` and `PartOfSpeech("13", "5", "small", "adjective")` as observations with belief value 1, since we know the parts of speech for these words for sure, but `PartOfSpeech("4", "4", "walk", "noun")` and `PartOfSpeech("4", "4", "walk", "verb")` as targets and let PSL infer the part of speech of "walk" in sentence 4.

The LINQS implementation additionally distinguishes *closed* and *open* predicates. Atoms of closed predicates are always observations with fixed belief values, while atoms of open predicates can be either targets or observations. These two types of predicates have further distinctive properties inside ground rules, as will be explained in section 2.2.2.

2.1.2 Rules

The dependencies between atoms of different predicates are specified as *rules*. Next to logical rules, it is possible to impose equality or inequality rules on the belief values of atoms that cannot be captured by first-order logic. Rules can be hard constraints or violable weighted rules.

2.1.2.1 Logical Rules

The most straightforward way of specifying PSL logical rules is the implication, the corresponding operator being `->` or `<-`. This rule, for instance, encodes the idea that if a word was of some part of speech in one sentence, it is likely that it has the same part of speech in another sentence:

```
PartOfSpeech(SentenceID1, Idx1, Word, PoS) -> PartOfSpeech(SentenceID2,
    Idx2, Word, PoS)
```

I refer to atom "templates" inside rules, like `PartOfSpeech(SentenceID1, Idx1, Word, PoS)`, as *literals*, since they refer to a specific restricted set of atoms.

The antecedent of an implication can also be a conjunction (`&` or `&&`) of several predicates, while the consequent may be a disjunction (`|` or `||`). The following two rules specify that if a word is preceded by a determiner, it is likely a noun, and if it is preceded by an adjective, it is likely a noun or another adjective:

```
PartOfSpeech(S, I1, W1, "determiner") & NextInteger(I1, I2) ->
    PartOfSpeech(S, I2, W2, "noun")
PartOfSpeech(S, I1, W1, "adjective") & NextInteger(I1, I2) ->
    PartOfSpeech(S, I2, W2, "noun") | PartOfSpeech(SentenceID, I2, W2,
    "adjective")
```

The antecedent of an implication may not contain a distinction. Likewise, conjunctions are disallowed in the consequents. The background of this is that internally, PSL operates only on purely disjunctive clauses. Before inference, implications therefore have to be converted into disjunctive clauses as well. The adjective rule above, for example, can be rewritten as (`~` or `!` being the negation operator):

```
    PartOfSpeech(S, I1, W1, "adjective") & NextInteger(I1, I2) -> PartOfSpeech(S,
    I2, W2, "noun") | PartOfSpeech(SentenceID, I2, W2, "adjective")
≡   ~(PartOfSpeech(S, I1, W1, "adjective") & NextInteger(I1, I2)) | PartOfSpeech(S,
    I2, W2, "noun") | PartOfSpeech(SentenceID, I2, W2, "adjective")
≡   ~PartOfSpeech(S, I1, W1, "adjective") | ~NextInteger(I1, I2) | PartOfSpeech(S,
    I2, W2, "noun") | PartOfSpeech(SentenceID, I2, W2, "adjective")
```

Logical rules can also be given in their disjunctive form directly.

2.1.2.2 Arithmetic Rules

Dependencies between atoms can also be modeled as *equality* (using operator `=`) or *inequality* (`<=` or `>=`) rules. In these *arithmetic rules*, the literals on each side of the (in-)equation are not connected by logical operators, but by the arithmetic operators `+` and `-` (plus `*` and `/` in combination with scalars). The following rule, for instance, specifies that the sum of all belief values for parts of speech "noun", "verb" and "adjective" of a single word must add up to 1, i.e. the model must choose exactly one of these parts of speech for a word:

```
PartOfSpeech(S, I, W, "noun") + PartOfSpeech(S, I, W, "verb") +
    PartOfSpeech(S, I, W, "adjective") = 1
```

Having to explicitly write out all parts of speech is rather inconvenient and often, it might not even be possible to enumerate all values, which is why there is a way to sum over all possible values a variable can take. In the following rule, `PartOfSpeech(S, I, W, +PoS)` is called the *summation atom*, since it sums over all existing atoms (observations and targets) with some constant inserted for the *sum variable* `PoS`:

```
PartOfSpeech(S, I, W, +PoS) = 1
```

The number of different constant insertions for a sum variable `SumV` can be referred to in a rule as the *cardinality* `|SumV|`. This can be used, for example, to extract the part of speech frequencies for each sentence:

```
PoSFrequency(PoS, Sentence) = PartOfSpeech(Sentence, +Idx, +Word, PoS)
    / |Idx|
```

Note that this only works when there are `PartOfSpeech` atoms for each word and possible part of speech (the implausible parts of speech then set to 0.0), since `PartOfSpeech(Sentence, +Idx, +Word, PoS)` will only sum over existing atoms. For example, let's say we only have the following atoms for the sentence "she saw him", because the parts of speech of "she" and "him" were not inferred over:

```
PartOfSpeech("1", "1", "she", "pronoun") = 1.0
PartOfSpeech("1", "2", "saw", "noun") = 0.0
PartOfSpeech("1", "2", "saw", "verb") = 1.0
PartOfSpeech("1", "3", "him", "pronoun") = 1.0
```

In this case, `PoSFrequency("verb", "1")` will have a belief of 1.0, because:

```
    PoSFrequency("verb", "1") = PartOfSpeech("verb", +Idx, +Word, "1") / |Idx|
≡   PoSFrequency(PoS, Sentence) = PartOfSpeech("1", "2", "saw", "verb") / 1
```

If we additionally had the atoms `PartOfSpeech("1", "1", "she", "verb")` and `PartOfSpeech("1", "3", "him", "verb")`, both set to 0.0, `PoSFrequency("verb", "1")` would get the correct belief value, $0.\overline{3}$, because:

```
    PoSFrequency("verb", "1") = PartOfSpeech("verb", +Idx, +Word, "1") / |Idx|
≡   PoSFrequency(PoS, Sentence) = (PartOfSpeech("1", "1", "she", "verb") +
    PartOfSpeech("1", "2", "saw", "verb") + PartOfSpeech("1", "3", "him", "verb"))
    / 3
≡   PoSFrequency(PoS, Sentence) = (0.0 + 1.0 + 0.0) / 3
```

Cardinalities in rules are therefore only of limited use when there is a large number of candidate targets for a predicate that you do not want to spell out completely for performance reasons.

Even when all combinations of constants exist for a summation atom, the number of matched atoms can be restricted by *filter clauses*, logical clauses applied to a sum variable. Only those constants that fulfill the filter clause can be inserted for the sum variable. For our `PoSFrequency`, we could, for example, exclude stop words:

```
PoSFrequency(PoS, Sentence) = PartOfSpeech(Sentence, +Idx, +Word, PoS)
    / |Idx| {Word: ~StopWord(Word)}
```

In the LINQS implementation, filter clauses are evaluated using hard logic, i.e. with truth values of either 0 or 1, where all non-zero belief values are rounded up to a truth value of 1. For this reason, both conjunctions and disjunctions can be used in the filter clause. Also, only literals of closed predicates may appear in a filter clause, but no explicit equations or inequations (like `SumV = "SomeValue"`). All variables appearing in the filter clause,

apart from the sum variable, must be non-sum variables also occurring in the associated arithmetic expression.

2.1.2.3 Rule Weight

By default, PSL rules are *unweighted rules*, also called *hard constraints*. They are always enforced and atoms are not allowed to violate them. To explicitly mark a rule as a constraint, one can append a dot to the rule:

```
PartOfSpeech(S, I, W, +PoS) = 1 .
```

In the LINQS implementation, constraints can sometimes be violated, for example when there are several contradictory constraints, which are not prevented or even reported by the system. It is therefore advised to keep the number of constraints low and manually check if any conflicts could arise between constraints to prevent unwanted effects.

Weighted rules, on the other hand, are explicitly allowed to be violated. The weight of a rule determines its precedence over other rules, or, put differently, how much a violation of this rule is penalized by the system. Weights can be any positive real number, where higher weights signify higher importance or higher penalization of violation. They can be specified explicitly followed by a colon at the beginning of a rule, as in:

```
2.0: PartOfSpeech(S, I1, W1, "determiner") & NextInteger(I1, I2) ->
    PartOfSpeech(S, I2, W2, "noun")
```

The hierarchy between differently weighted rules is rather strict by default, leading to a "winner take all" situation in the sense that the highest weighted rule is attempted to be satisfied before lower weighted rules are considered. The reason for this lies in the mathematical model behind PSL, which is explained in section 2.3. To achieve a smoother hierarchy, `^2` (the significance of which will also be covered in section 2.3) can be appended to a weighted rule:

```
2.0: PartOfSpeech(S, I1, W1, "determiner") & NextInteger(I1, I2) ->
    PartOfSpeech(S, I2, W2, "noun") ^2
```

The LINQS implementation comes with several learning algorithms for learning weights from gold standard data. I am working with manually tuned weights in my own model, however.

2.2 The LINQS Grounding Process

Grounding is the process of substituting the literals in rules with matching ground atoms and converting logical rules into disjunctive form. A *ground rule* then is a rule that contains only ground atoms. Before inference, all rules have to be grounded. An example for a grounding of the determiner-noun rule is:

```
2.0: ~PartOfSpeech("1", "4", "the", "determiner") | ~NextInteger("4",
    "5") | PartOfSpeech("1", "5", "tree", "noun") ^2
```

Summing atoms are also spelled out explicitly. If the only part of speech candidates for "tree" were "noun" and "verb", a grounding for the rule `PartOfSpeech(S, I, W, +PoS) = 1 .` would be:

```
PartOfSpeech("1", "5", "tree", "noun") + PartOfSpeech("1", "5", "tree",
    "noun") = 1 .
```

Understanding the grounding process of the LINQS implementation is important for writing rules that behave as intended. This section therefore discusses how LINQS assembles the ground rules and the sometimes puzzling behavior that results from it, as well as some design patterns the EtInEn group has developed to facilitate the development of large PSL models.

2.2.1 Grounding Variables

Consider the following rule:

`~P1(A, B) & P2(A, C) -> P3(B, C)` ≡ `P1(A, B) | ~P2(A, C) | P3(B, C)`

LINQS will throw an error when adding this rule to the model:

```
IllegalArgumentException: Any variable used in a negated
    (non-functional) predicate must also participate in a positive
    (non-functional) predicate. The following variables do not meet
    this requirement: [B].
```

This error message is misleading, because it refers to the *negated* disjunctive normal form (nDNF) of the rule, which is the conjunction `~(P1(A, B) | ~P2(A, C) | P3(B, C))` ≡ `~P1(A, B) & P2(A, C) & ~P3(B, C)`. Here, `B` occurs only in negated literals. The following rules would both be fine, since `B` is now covered by `P4`:

`~P1(A, B) & P2(A, C) & P4(B) -> P3(B, C)`
`~P1(A, B) & P2(A, C) -> P3(B, C) | ~P4(B)`
$\Big\}\ \tilde{\rightarrow}$ `~P1(A, B) & P2(A, C) & ~P3(B, C) & P4(B)`

To ground a rule, LINQS retrieves all valid combinations of ground atoms from its atom database that can be inserted for the *positive* literals in the *nDNF* of the rule. In case of the above rule(s), for example, LINQS will query its database for all possible atom pairs (`P2(A, C)`, `P4(B)`). For each such tuple of ground atoms, a ground rule is created where the constants of the retrieved atoms are substituted for the variables occurring in both positive and negative literals of the rule. If the system has, for instance, found the pair (`P2("foo", "bar")`, `P4("baz")`) in the database, yielding the variable assignments `A = "foo"`, `B = "baz"` and `C = "bar"`, it will create the ground rule:

`~P1("foo", "baz") & P2("foo", "bar") & ~P3("baz", "bar") & P4("baz")` ≡ `P1("foo", "baz") | ~P2("foo", "bar") | P3("baz", "bar") | ~P4("baz")`

This is the reason for the requirement that every argument must appear in a positive literal of the nDNF: If one does not, there is no ground atom containing it retrieved from the database and the system does therefore not know its value.

To circumvent this restriction, the EtInEn group has introduced the concept of *existential predicates*. These are closed predicates that match the arguments of another predicate that may occur only negated in some rule. Whenever an atom for the main predicate is inserted into the database, an atom for the existential predicate with the exact same arguments is inserted as well with belief 1.0. Existential predicates can be read as "there exists an atom for predicate <main> with these arguments". Consider the determiner-noun rule again:

```
PartOfSpeech(S, I1, W1, "determiner") & NextInteger(I1, I2) ->
    PartOfSpeech(S, I2, W2, "noun")
```

This rule will actually be rejected by LINQS, because the argument `W2` only occurs in a negated literal in the nDNF. To resolve this, we introduce an existential predicate `ExistsPartOfSpeech` and rewrite the rule as:

```
PartOfSpeech(S, I1, W1, "determiner") & NextInteger(I1, I2) &
    ExistsPartOfSpeech(S, I2, W2, "noun") -> PartOfSpeech(S, I2, W2,
    "noun")
```

All arguments of the negated `PartOfSpeech(S, I2, W2, "noun")` are now matched by a positive literal in the nDNF and the rule can be successfully grounded.

2.2.2 Grounding Open and Closed Predicates

That LINQS requires all arguments to occur in positive literals in the nDNF may seem like a limitation, but it can also be a feature. Remember that it retrieves only the positive atoms from the database and generates all negative atoms on the fly without checking if they were actually entered by the user. The following rule states that a symbol which is pronounced unvoiced, alveolar and as a stop must correspond to the phoneme [t]:

```
~IsVoiced(Symbol) & HasPlace(Symbol, "alveolar") & HasManner(Symbol,
    "stop") -> IsPronouncedAs(Symbol, "t")
```

Let's assume `IsPronouncedAs` is open while the other predicates are closed, and we have entered the following atoms:

```
HasPlace("ᴛ", "alveolar") = 1.0
HasManner("ᴛ", "stop") = 1.0
IsPronouncedAs("ᴛ", "d")
IsPronouncedAs("ᴛ", "t")
```

No atom has been entered for `IsVoiced`. However, `IsVoiced` and `IsPronouncedAs` are negated in the nDNF of this rule, so their atoms are not retrieved from the database during grounding. LINQS is therefore able to generate the following ground rule and run an inference on it:

```
~IsVoiced("ᴛ") & HasPlace("ᴛ", "alveolar") & HasManner("ᴛ", "stop") ->
    IsPronouncedAs("ᴛ", "t")
```

The non-existing `IsVoiced("ᴛ")` is treated as having belief value 0.0 and the system correctly concludes that `IsPronouncedAs("ᴛ", "t") = 1.0`. However, when we declare `IsVoiced` as an open predicate, e.g. because the more general phonetic properties of the symbols are also supposed to be inferred, we are presented with an error message after grounding:

```
PersistedAccessException: Can only call getAtom() on persisted
    RandomVariableAtoms (RVAs) using a PersistedAtomManager. Cannot
    access ISVOICED('ᴛ'). This typically means that provided data is
    insufficient. An RVA (atom to be inferred (target)) was constructed
    during grounding that does not exist in the provided data.
```

The grounding was successful, but the automatically generated atom `IsVoiced("ᴛ")` does not actually exist in the database. This poses a problem when the atom's predicate is open, because this means that its belief value can be changed during inference. The new belief value must be written back into the database, which is impossible when it does not contain the atom, since PSL is not allowed to create atoms on its own. For closed predicates, this is not problematic, because their belief values cannot change anyway, so they are never attempted to be written back.

Existential predicates partially solve this problem as well. We can rewrite the rule as follows:

```
ExistsIsVoiced("ᴛ") & ~IsVoiced("ᴛ") & HasPlace("ᴛ", "alveolar") &
    HasManner("ᴛ", "stop") -> IsPronouncedAs("ᴛ", "t")
```

LINQS no longer grounds rules with non-existing `IsVoiced` atoms. This means that the error above is no longer thrown, but it also means that this rule does not influence the belief value of `IsPronouncedAs("ᴛ", "t")` anymore. In the worst case, we end up with no evidence for or against `IsPronouncedAs("ᴛ", "t")` at all. Wherever feasible, one should therefore insert all possible atoms for an open predicate and fix them to 0.0 belief if necessary rather than omit them.

2.2.3 Priors as Replacement for Negative Evidence

Unfortunately, it is not always viable to spell out all possible atoms for a predicate. Consider the following two rules:

```
1.0: Trigram(Left, Target1, Right) & Trigram(Left, Target2, Right) ->
    SimilarContexts(Target1, Target2)
1.0: Trigram(Left, Target1, Right) & ~Trigram(Left, Target2, Right) ->
    ~SimilarContexts(Target1, Target2)
```

These two rules model whether two words occur in similar contexts. The first rule provides positive evidence, boosting `SimilarContexts` whenever the two target words are observed in the same contexts, and the second rule provides negative evidence, weakening `SimilarContexts` whenever there is an unshared context.

It is not feasible to have 0.0 belief `Trigram`s for every possible combination of three words. Usually, you would only want to insert those trigrams that are actually observed. As long as `Trigram` is a closed predicate though, these rules do their work, since the negated `Trigram` is filled in during grounding even though it does not exist as an actual atom[1]. However, this does not happen if `Trigram` is an open predicate because its belief is inferred in PSL as well. In this case, the second rule never applies, and since there is no counter-evidence, any two words that occur in the same contexts at least once will have a `SimilarContexts` of 1.0.

There is no fully satisfying solution to this problem. A workaround that does not involve the mass creation of 0.0 belief atoms are negative priors, weighted rules that keep target atoms down by default until enough positive evidence has been observed to assign them a higher belief value. This is such a negative prior for `SimilarContexts`:

```
5.0: ~SimilarContexts(Target1, Target2)
```

This rule unconditionally negates `SimilarContexts` with a high weight, which ensures that a single match of the positive rule from before is not sufficient to raise its belief. Only when the combined weights of the rules providing positive evidence surpass the negative prior will they be able to push `SimilarContexts`s belief up.

The weight of the negative prior must be chosen carefully: If it is set too low, it will have no effect; if it is set too high, `SimilarContexts` atoms will rarely have a belief above 0.0. There is also the danger that the height of the prior's weight may be input-dependent: Some input data may generate lots of positive evidence, hitting the threshold easily, while other input data could provide too sparse evidence to ever overcome the negative prior. When putting this technique into use, one should take care that varying quality of input data does not alter the effect of the negative priors.

2.3 Hinge-Loss Markov Random Fields

Mathematically, a PSL model with its atoms and rules is a *hinge-loss Markov random field* (HL-MRF). In general, a *Markov random field* (MRF) is a probabilistic graphical model that assigns probability mass using weighted feature functions, the potentials. HL-MRFs are a subclass of MRFs whose potentials are hinge losses, i.e. functions consisting of a constant and a linear section (Bach et al. 2017).

2.3.1 Translating Atoms and Rules into HL-MRFs

Let $y = (y_1, \ldots, y_n)$ be a vector of real-valued random variables from the interval $[0, 1]$, $\tilde{D} \subseteq [0, 1]^n$ the set of feasible variable vectors, $\phi = (\phi_1, \ldots, \phi_m)$ a vector of potentials and $w = (w_1, \ldots, w_m)$ a vector of real-valued weights. The inference objective of an HL-MRF then is:

$$\arg\min_{y \in \tilde{D}} \sum_{j=0}^{m} w_j \phi_j(y) \tag{2.1}$$

[1]Note that this can be problematic as well, because the number of ground rules the inference is run on has great influence on performance.

In PSL terms, y are the belief values of the n ground atoms in our atom database, ϕ the degrees of violation of our m (weighted) ground rules and w the weights of these rules. Informally, equation (2.1) states that the inference result is the assignment of belief values to our atoms that least violate our rules.

The potential ϕ_j is calculated as

$$\phi_j(y) = \max\{\ell_j(y), 0\} \tag{2.2}$$

where ℓ_j is a linear function. Informally, ℓ_j is the *distance to satisfaction* of our ground rule, i.e. how far it is from being satisfied. $\ell_j < 0$ indicates that the ground rule is oversatisfied, whereas $\ell_j > 0$ means that it is violated by the belief values assigned in y.

Due to the piecewise linear nature of the hinge loss potentials, conflicting rules (i.e. distance to satisfaction functions assigning complementary values to the same y) can lead to the aforementioned "winner take all" situation: When they have different weights, it is cheaper to always fully satisfy the rule with higher weight because its term overrules the term of the lower-weighted rule, which is then always completely violated. To achieve a "smoother" result, an optimal y that attempts to partially satisfy both rules, the potentials can be squared (cf. Bach et al. 2017, p. 11f.). However, in my experience, this will often make more complex systems rather undecisive, because it shifts the optimal belief value towards the middle of the belief spectrum in order to give credit to all rules, yielding many atoms with inconclusive beliefs around the 0.5 mark. Fortunately, the potential squaring can be switched on manually during rule specification by appending `^2` to the rule string, as was already described in section 2.1.2.3.

The distance to satisfaction of a logical ground rule is calculated as

$$\ell_j(y) = 1 - \sum_{i \in I_j^+} y_i - \sum_{i \in I_j^-} (1 - y_i) \tag{2.3}$$

where I_j^+ are the indices of those atoms that occur positively in the ground rule and I_j^- are the indices of those atoms that occur negated.

The real-valued truth values of our logical ground rules are calculated using Łukasiewicz logic, where the conjunction, disjunction and negation operators $\wedge$, $\vee$ and $\neg$ are defined as:

$$p \wedge q = \max\{p + q - 1, 0\}^2 \tag{2.4}$$

$$p \vee q = \min\{p + q, 1\} \tag{2.5}$$

$$\neg p = 1 - p \tag{2.6}$$

[2]Beltagy, Erk, and Mooney (2014) found the Łukasiewicz conjunction to be too restrictive when applying to multiple target atoms with belief < 1.0: For $p = q = 0.5$, for instance, the conjunction $p \wedge q$ is evaluated to 0.0, even though both atoms are partially believed to be true. They reformulate conjunction as the averaging function $p \wedge q = \frac{p+q}{2}$, a modification also adopted by Liu et al. (2016). In my model, conjunctions typically operate on at most one or two target atoms, for which the Łukasiewicz conjunction is not too problematic.

The distance to satisfaction of a Łukasiewicz disjunction would be how far its value is from its maximum value 1, namely $1 - \min\{p + q, 1\} = \max\{1 - p - q, 0\}$, which directly results in the potential ϕ_j in (2.2) with the ℓ_j from (2.3) inserted.

The distance to satisfaction of an inequality rule $l_j(y) \leq g_j(y)$ can be straightforwardly computed as:

$$\ell_j(y) = l_j(y) - g_j(y) \tag{2.7}$$

An equality rule $t_{j_1}(y) = t_{j_2}(y)$ is translated into two inequality rules $t_{j_1}(y) \leq t_{j_2}(y)$ and $t_{j_1}(y) \geq t_{j_2}(y)$.

Finally, the feasible set $\tilde{D}$ is defined as

$$\tilde{D} = \{y \in [0, 1]^n \mid \forall c_k \in c : c_k(y) \leq 0\} \tag{2.8}$$

where $c = (c_1, \ldots, c_r)$ is the vector of distance to satisfaction functions of the hard constraints, the rules that may not be violated.

2.3.2 Properties of HL-MRFs

The problem of solving the inference objective (2.1), i.e. finding the most probable assignment of belief values y, is an instance of *maximum a posteriori* (MAP) inference. MAP inference on discrete truth values is equivalent to the MAX SAT problem, which can be approximated using randomized algorithms that relax MAX SAT to a convex program (Bach et al. 2017). While these relaxation techniques guarantee high-quality solutions, they scale poorly to large graphical models such as MRFs. On the other hand, discrete MRFs can be approximated by applying *local consistency relaxation* (LCR), which is highly scalable but does not guarantee the quality of the result. Bach et al. (2017) do not only prove that MAX SAT relaxation and LCR "solve equivalent optimization problems with identical solutions" (p. 5), but also that this equivalence holds for MAP inference on both discrete and continuous truth values when interpreted using Łukasiewicz logic. This implies that algorithms developed for all three tasks can be combined for scalable and high-quality reasoning about HL-MRFs.

To exploit the typically sparse dependency structure of HL-MRFs, the MAP algorithm developed by Bach et al. (2017) uses *consensus optimization*, a technique that splits the optimization problem into smaller subproblems using the *alternating direction method of multipliers* (ADMM), solves them independently, and tries to find a consensus solution from the individual subsolutions. This technique is not only very efficient but also allows for easy parallelization. The inference can be sped up further by *lazy MAP inference*, i.e. dropping some highly satisfied potentials or constraints, which can, however, have a negative impact on the quality of the results.

Comparing their MAP inference algorithm to the commercial convex optimization toolkit MOSEK, which uses the popular interior-point methods (IPMs), Bach et al. (2017, p. 33ff.) found that their optimization technique scaled extremely well to larger problems. While the IPMs' running time quickly exploded, ADMM turned out to scale linearly with the size of the problem, running only 70 seconds on 397,000 potentials and constraints of both

piecewise-linear and piecewise-quadratic MAP problems. IPM, on the other hand, took 37 minutes on the same size when solving a piecewise-linear MAP problem, and could not be tested on piecewise-quadratic MAP problems larger than 225,000 potentials and constraints, where it ran for more than 6 hours. The high performance of the ADMM method comes at the cost of lower-quality results; however, the relative error ranges only between 0.2% and 0.4% (Bach et al. 2017, p. 36).

HL-MRFs therefore allow for fast and accurate inference of continuous truth values constrained by first-order logic rules. The MAP inference algorithm developed by Bach et al. (2017) for HL-MRFs is tailored towards the sparse dependency structure of logical rules, scales extremely well to large problems and can be readily parallelized. This makes them a good fit for solving computational problems that can naturally be formulated as a set of first-order logical rules.

2.4 PSL for Historical Linguistics

PSL has already been put into use for solving various problems in computational linguistics, such as semantic similarity judgment (Beltagy, Erk, and Mooney 2014; Beltagy 2016), recognizing textual entailment (Beltagy 2016; Wang and Ku 2016; Wang and Ku 2017), question answering (Beltagy 2016), event classification (Liu et al. 2016), named entity recognition and entity linking (Rospocher 2018), anaphora resolution (Prakash et al. 2019), sentiment analysis (Deng and Wiebe 2015), and stance classification (Sridhar et al. 2015). All of these tasks lend themselves naturally to PSL, since they typically involve advanced reasoning and weighting of different types of evidence. Comparative historical linguistics, despite having been overlooked so far by the PSL community, is another natural use case for PSL. As Crowley and Bowern (2010) note:

> "[T]he comparative method is not an algorithm for 'discovering' protolanguages; rather, it is a set of heuristics (guiding tools) for you to use in making hypotheses about the past history of languages." (p. 162)

The individual steps of the comparative method are highly interconnected, each providing more (counter-)evidence for all of the others. There are always competing observations and principles weighing in: The members of a sound correspondence provide evidence for one proto-phoneme, a common sound change supports another, and both violate the symmetric inventory principle; the occurrences of a regular sound correspondence support a certain conditioned sound law, whose context, however, overlaps with a different sound law for the same proto-phoneme, and either would make a cognate set implausible that provides evidence for another sound correspondence... These kinds of dependencies are difficult to model computationally, but seem predestined for being formulated inside PSL.

While other probabilistic models, especially when machine learning is involved, often present themselves as a "black box" whose reasoning is hard to retrace, PSL models can explain themselves in an understandable way, since the rules they operate on are modeled after the same principles human linguists use for reasoning. This should not only make a PSL model easier to debug, but can also lead to interesting insights as to how much the different parts of the comparative method each influence the outcome.

3 Preparation of Gold Standard Sound Law Sets

In order to evaluate a system for automated sound law inference, one needs a complete set of gold standard sound laws to compare the system's output against. Unfortunately, no such gold standard exists yet to my knowledge. Bouchard-Côté et al. (2013), despite also inferring individual sound laws, only evaluate their output against full proto-word reconstructions. List (2019a) does not directly evaluate the detected sound correspondence patterns, but rather uses them to predict unseen cognates and compares these against the data. Hruschka et al. (2015) have a gold standard set of sound laws for Turkic, but do not give contexts for these because their system can only detect unconditioned sound change. Since SoInEn, the system presented in this thesis, is designed to infer conditioned sound laws as well, I cannot use their gold standard.

I therefore compiled my own two gold standard sound law sets for some of the Dravidian and Samoyedic languages, which I present in this chapter. Dravidian was chosen due to my familiarity with some of its members (Malayāḷaṁ and, to some extent, Kannaḍa) and is a rather simple test case, with the selected daughter languages still being rather close to each other. The Samoyedic languages underwent more drastic sound changes and serve as a difficult test case to see how the system performs on a rather obscure test set. Initially, I intended to compile gold standards for several more language families, but this turned out to me a more tedious task than expected, since the information about sound change is often spread across many sources for each descendant language, and there is usually not one comprehensive survey of all the sound changes that happened between a proto-language and its children. In this respect, Dravidian and Samoyedic also had the advantage of having several comparative sources compiling the phonetic history of the entire family that I could rely on.

3.1 Dravidian

The members of the Dravidian language family are spoken in India, primarily in the southern states, and some bordering countries. They can be roughly divided into four groups: 1) North Dravidian, comprising Brahui, which is spoken in Pakistan, as well as Kuṛukh

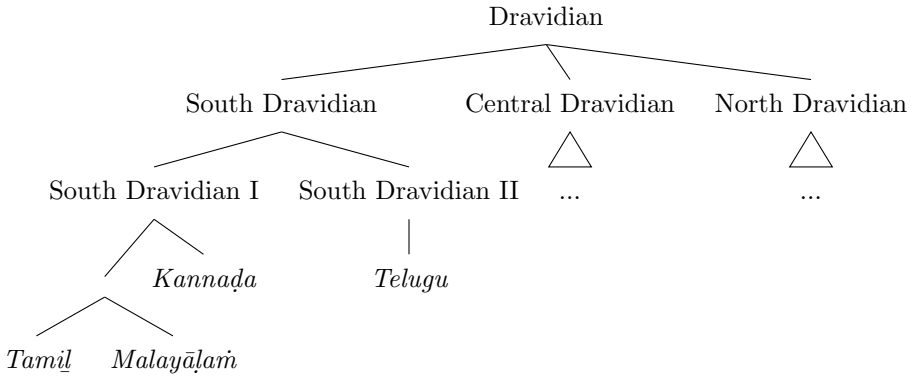

Figure 3.1: The South Dravidian languages within the Dravidian language family tree.

	Front		Central		Back	
High	*i* i	*ī* iː			*u* u	*ū* uː
Mid	*e* e	*ē* eː			*o* o	*ō* oː
Low			*a* a	*ā* aː		

Table 3.1: The reconstructed vowel inventory of Proto-Dravidian (the usual romanization in italics to the left, IPA to the right).

and Malto, which are spoken in Nepal and the north-eastern part of India, 2) Central Dravidian, which contains a number of languages spoken in small areas in central India, and 3) South Dravidian, which is the largest group and comprises most of the Dravidian languages spoken in southern India, among them the four literary languages Tamiḻ, Malayāḷaṁ, Kannaḍa and Telugu (Krishnamurti 2003). South Dravidian can further be divided into South Dravidian I and II, with Tamiḻ, Malayāḷaṁ and Kannaḍa belonging to the South Dravidian I and Telugu to the South Dravidian II group (Krishnamurti 2003). Figure 3.1 illustrates the position of the four languages in the Dravidian language family.

Of the Dravidian languages, the NorthEuraLex database only contains Tamiḻ, Malayāḷaṁ, Kannaḍa and Telugu, which is why I focus on these four languages in the following sections. In particular, I only describe the sound laws that applied to these languages, so I am not covering the evolution of North and Central Dravidian. All four are official languages in their respective home states with long literary traditions and are used in modern everyday speech and writing by tens of millions of native speakers. As such, they are still actively evolving and particularly well-documented.

3.1.1 Proto Vowels

There is broad agreement that Proto-Dravidian had ten vowels, namely the members of the standard five-vowel system (/a/, /e/, /i/, /o/ and /u/) as both short and long vowels (Andronov 2003; Burrow and Emeneau 1984; Krishnamurti 2003). Table 3.1 shows the resulting vowel space for Proto-Dravidian. The five vowel qualities have been preserved in all modern Dravidian languages and the length contrast was retained in all except Brahui,

	Labial	Dental	Alveolar	Retroflex	Palatal	Velar
Stop/Affric.	*p* p	*t* t̪	*ṯ* t	*ṭ* ʈ	*c* t͡ɕ	*k* k
Nasal	*m* m	*n* n̪		*ṇ* ɳ	*ñ* ɲ	
Lateral			*l* l	*ḷ* ɭ		
Flap			*r* ɾ			
Approximant	*v* ʋ			*ẓ/ḻ* ɻ	*y* j	

Table 3.2: The reconstructed consonant inventory of Proto-Dravidian (the usual romanization in italics to the left, IPA to the right).

where it was lost only for /e/ and /o/ (Krishnamurti 2003).

In the orthography of modern Dravidian languages, the sequences /ai/ and /au/ are usually treated as diphthongs. From a phonological point of view, however, they are better analyzed as vowel-glide sequences /aj/ and /aw/, respectively: Phonotactically, they pattern with other vowel-consonant rhymes where the coda is a sonorant (Krishnamurti 2003, pp. 118f.). Hence, Proto-Dravidian is believed to not have had any diphthongs (Andronov 2003; Burrow and Emeneau 1984; Krishnamurti 2003).

3.1.2 Proto Consonants

The situation is slightly less clear when it comes to the consonant inventory of Proto-Dravidian. Table 3.2 shows the inventory that I will assume in this thesis.

It is generally agreed that there Proto-Dravidian had no voicing contrast in plosives and affricates (Andronov 2003; Burrow and Emeneau 1984; Krishnamurti 2003) and Krishnamurti (2003, pp. 132ff.) shows in a quantitative study that even in modern Dravidian languages the voiceless stops are dominating. Instead, Proto-Dravidian contrasted geminate and non-geminate stops, which has developed into a voicing contrast in many modern Dravidian languages (Andronov 2003; Burrow and Emeneau 1984; Krishnamurti 2003).

Proto-Dravidian is noted for having a three-way contrast in apical stops, distinguishing dental /t̪/, alveolar /t/ and retroflex /ʈ/. The alveolar stop has developed into an alveolar trill /r/ in most modern Dravidian languages and is thus also often written as *ṟ*.

The phonemic status of the alveolar nasal /n/ in contrast to the dental nasal /n̪/, on the other hand, is debated. Burrow and Emeneau (1984) list them as allophones, but note that "Ta. [Tamiḻ] and Ma. [Malayāḷaṁ] seem to have evidence for two phonemes [...] in PDr. [Proto-Dravidian]" (p. xiii). According to Andronov (2003, p. 83) and Krishnamurti (2003, p. 138), the two nasals contrasted in several words of Classical Tamiḻ in intervocalic or word-final position. In general, however, the dental nasal occurred word-initially and before dental consonants, while the alveolar nasal dominated in all other positions. Since the alveolar nasal has completely merged into the dental nasal in many modern Dravidian languages, historical linguists usually only reconstruct /n̪/ as a phoneme and treat /n/ as a positional allophone, which is why I have not included it in the Proto-Dravidian inventory for this thesis.

Another peculiar case is the palatal nasal /ɲ/, which is extremely common in modern Dravidian languages, but often only occurs as an allophone of /n̪,n/ next to a palatal consonant (Andronov 2003, p. 32). It has phonemic status in some languages though, notably Malayāḷaṁ and Tamiḻ, where it also occurs word-initially (Krishnamurti 2003, pp. 139ff.). Due to its frequent co-occurrence with other palatals, Burrow and Emeneau (1984) does not list /ɲ/ as a Proto-Dravidian consonant and Andronov (2003, p. 83) reconstructs Tamiḻ and Malayāḷaṁ /ɲ/ as having developed from Proto-Dravidian /n̪,n/. He notes, however, that there exists an alternation of word-initial /na/, /ne/ in several words of some Dravidian languages that could be reconstructed as stemming from Proto-Dravidian /na/, /ɲa/, indicating that /ɲ/ was in fact in the Proto inventory. Krishnamurti (2003, pp. 139ff.) also presents /ɲ/ as a Proto-Dravidian phoneme occurring only word-initally, mostly before /a/ and /e/, that developed into /n̪,n/ in most daughter languages. I adopt this view and regard /ɲ/ as a proto-consonant.

There are different opinions on the phonetic features of Proto-Dravidian /v/. Most authors use the symbol *v* for it. Andronov (2003, pp. 32f.) notes that both bilabial [w] and labio-dental [v] occur among the Dravidian languages, but that "[t]he labio-dental phoneme is predominant". Krishnamurti (2003, p. 141f.), however, believes that the Proto-Dravidian phoneme was actually bilabial [w], because no rounded vowels (i.e. /u/, /o/) occur after it, and claims that "[s]everal authors write <v> instead but they do not mean that the sound that they are representing is labio-dental" (p. 142). In this thesis, I assume /v/ to have been [ʋ], because that is the IPA symbol used in the NorthEuraLex transcriptions of the Dravidian languages. Phonetically, it is also in between [v] and [w], the suggestions of Andronov (2003) and Krishnamurti (2003).

Most sources do not reconstruct any fricatives for Proto-Dravidian (apart from the debatable [v]). Andronov (2003) lists /s/ as a proto-phoneme, but it appears only intervocally in place of /t͡ɕ/. Since /s/ is a common reflex of */t͡ɕ/ in Kannaḍa and Telugu (Krishnamurti 2003; Burrow and Emeneau 1984) and an allophone of /t͡ɕ/ in modern Tamiḻ (Keane 2004) between vowels, I do not consider it a separate phoneme of Proto-Dravidian.

Krishnamurti (2003) proposes an additional phoneme *H* which he characterizes as a glottal or laryngeal non-sonorant glide that did not occur word-initially (pp. 91, 93). This suggestion is based on a phoneme *ḥ* called *āytam* that appeared in a few words in Early Tamiḻ and assimilated to the following stop (resulting in a geminate). According to Krishnamurti (2003, pp. 154ff.), *H* explains several words that alternate between long and short vowels in different inflectional forms, as well as the unexpected appearance of /h/ and /j/ in some words of several Dravidian languages. Since the evidence for *H* is so small and Krishnamurti is the only author proposing it, I have decided to not include *H* in the Proto-Dravidian inventory for this thesis.

3.1.3 Phonotactics

In general, Proto-Dravidian had a simple (C)V(S) syllable structure (S = sonorant). Just the initial syllable could display the full range of vowels, whereas in later syllables, only the vowels [a], [i] and [u] are found (Krishnamurti 2003, pp. 90ff.). The consonants [t], [ɾ] and all retroflex sounds did not appear word-initially (Krishnamurti 2003, p. 120). Intervocalic

[p] had already been spirantized into [ʋ] in Proto-Dravidian and is therefore also not found (Krishnamurti 2003, p. 144).

All of the stops and the affricate could be geminate in Proto-Dravidian (Andronov 2003; Burrow and Emeneau 1984; Krishnamurti 2003). There are a few contrasting pairs for short and long /l/ and /ɭ/, indicating that they might have had geminate counterparts in the proto-language, but there is no evidence for such a distinction for the remaining sonorants (Krishnamurti 2003, p. 166). The only consonant clusters of Proto-Dravidian were word-medial homorganic nasal+stop clusters, which were possible with both non-geminate and geminate stops (Andronov 2003; Burrow and Emeneau 1984; Krishnamurti 2003).

The exact phonetic quality of the single and geminate stops is not entirely clear. Since most of the modern Dravidian languages show some degree of lenition or voicing of single plosives in intervocalic position and after the homorganic nasal, it has been proposed that this was already in effect in Proto-Dravidian (Krishnamurti 2003, p. 144; Andronov 2003, pp. 23f.). This is especially true for the South Dravidian languages, which all voice single stops intervocally and in nasal+stop clusters, so I will reconstruct these stops as voiced in the following sound laws.

3.1.4 Sound Changes

In this section I present the sound changes that occurred between Proto-Dravidian and Tamiḻ, Malayāḷaṁ, Kannaḍa and Telugu. $^{(\sim)}$ indicates an irregular sound change and $^{(\star)}$ points to a comment below the respective table. The first row of each table refers to the sources, with the respective page numbers in subscript. The two sources I am using are the comparative grammars by Andronov (2003), abbreviated AN, and Krishnamurti (2003), abbreviated KR.

3.1.4.1 Vowels

The Dravidian vowels are astonishingly stable in the South Dravidian languages. The only changes they went through are what is referred to by Krishnamurti (2003) as the "South Dravidian umlaut" (p. 101) and the "Kannaḍa umlaut" (p. 106). Here, in a first step, the high vowels [i(ː)] and [u(ː)] were lowered to [e(ː)] and [o(ː)], respectively, in all four languages when the nucleus of the following syllable was [a]. Then, however, [e(ː)] and [o(ː)] were raised again in the same environment in Tamiḻ and Malayāḷaṁ. Finally, Kannaḍa also raised [e(ː)] and [o(ː)] when followed by high vowels.

	Tamiḻ	**Malayāḷaṁ**	**Kannaḍa**	**Telugu**
***V**	$AN_{51,67ff}$,KR_{101ff}			
i(ː)			e(ː) | _ C* a	e(ː) | _ C* a
u(ː)			o(ː) | _ C* a	o(ː) | _ C* a
e(ː)	i(ː) | _ C* a	i(ː) | _ C* a	i(ː) | _ C* [HIGH]	
o(ː)	u(ː) | _ C* a	u(ː) | _ C* a	u(ː) | _ C* [HIGH]	

3.1.4.2 Obstruents

If one assumes the voicing opposition of word-internal non-geminate and geminate stops to have existed already in Proto-Dravidian, the obstruents are also very stable, with the exception of alveolar [t], which only survives in some contexts in Tamiḻ and Malayāḷaṁ, and the affricate [t͡ɕ] which underwent a number of irregular changes, not all of which I will discuss due to their limited occurrence.

	Tamiḻ	**Malayāḷaṁ**	**Kannaḍa**	**Telugu**
***p**	$AN_{54f,81f}$,$KR_{120f,163ff}$			
p-	p	p	h	p
mb			ʊ | Vː _	m | Vː _
	mb	mb	mb	mː
pː			p | Vː _	p | Vː _
	pː	pː	pː	pː
mpː	pː	pː	mp	mp
***t̪**	$AN_{47f,79ff}$,$KR_{145f,163ff}$			
t̪-	t̪	t̪	t̪	t̪
-d̪-	ð	d̪	d̪	d̪
n̪d̪	n̪d̪	n̪ː	n̪d̪	n̪d̪
t̪ː			t̪ | Vː _	t̪ | Vː _
	t̪ː	t̪ː	t̪ː	t̪ː
n̪t̪ː	t̪ː	t̪ː	n̪t̪	n̪t̪
***t**	$AN_{47f,77ff}$,$KR_{146,163ff}$			
-d-	r	r	r	r
nd	ɳː	n̪ː	n̪d̪	ɳɖ
tː			r | Vː _ (⋆)	ʈ | Vː _ (⋆)
	tː	tː	t̪ː	ʈː
ntː	tː	tː	n̪t̪	ɳʈ

(⋆) In Kannaḍa and Telugu, geminate stops were shortened to single stops after a long vowel (Krishnamurti 2003, pp. 163f.). For the geminate alveolar stop /tː/, it can be observed that Kannaḍa shows short /r/ and Telugu short /ʈ/ where Tamiḻ and Malayāḷaṁ have geminate /tː/ after long vowels:

(1) a. Tamiḻ *cāṟṟu*, Malayāḷaṁ *cāṟṟuka*, Kannaḍa *sāṟu*, Telugu *cāṭu* 'to publish, announce' (Burrow and Emeneau 1984)

b. Tamiḻ *ēṟṟu*, Malayāḷaṁ *ēṟṟuka*, Kannaḍa *ēṟisu* 'to raise', Telugu *ēṭavālu* 'slope' (Burrow and Emeneau 1984)

c. Tamiḻ *ūṟṟam*, Kannaḍa *ūṟu-gōlu* 'walking-stick', Malayāḷaṁ *ūṟṟam*, Telugu *ūṭa* 'strength' (Burrow and Emeneau 1984)

It seems that the Telugu law tː > ʈː applied before this sound change, but the Kannaḍa law tː > ʈ̪ː applied afterwards, because otherwise, we would observe short /ʈ̪/ in the Kannaḍa words above.

	Tamiḻ	**Malayāḷaṁ**	**Kannaḍa**	**Telugu**
***t͡ɕ**	AN$_{47f,74ff,88}$,KR$_{121ff,148f,163ff}$			
t͡ɕ-	∅ (~)(⋆)	∅ (~)(⋆)	∅ (~)(⋆)	∅ (~)(⋆)
	t͡ɕ	t͡ɕ	s	t͡ɕ
-d͡ʑ-	s	d͡ʑ	s	s
ɲd͡ʑ	ɲd͡ʑ	ɲː	ɲd͡ʑ	ɲd͡ʑ
t͡ɕː			t͡ɕ | Vː _	t͡ɕ | Vː _
	t͡ɕː	t͡ɕː	t͡ɕː	t͡ɕː
ɲt͡ɕː	t͡ɕː	t͡ɕː	ɲt͡ɕ	ɲt͡ɕ

(⋆) Word-initial /t͡ɕ/ has undergone a number of irregular sound changes in many of the Dravidian languages, notably all South Dravidian languages. Krishnamurti (2003, p. 121ff.) assumes a gradual weakening /t͡ɕ/ → /s/ → /h/ → ∅ that reached different stages in the different languages. It is irregular in that some words with initial /t͡ɕ/ retain it and some weaken or lose it. Krishnamurti (2003, p. 122) claims that only 14 % of words with Proto-Dravidian initial /t͡ɕ/ went through the gradual loss. In Burrow and Emeneau (1984), the forms with and without /t͡ɕ/ often co-occur and even within our NorthEuraLex 0.9 (NEL) database, /t͡ɕ/ shows different degrees of loss:

(2) a. 'salt': Tamiḻ *uppu* (NEL [upːu]), Malayāḷaṁ *uppu* (NEL [upːɪ̈]), Kannaḍa *uppu* (NEL [wʊpːu]), Telugu *uppu* (NEL [upːu]); but e.g. Kolami *sup*, Parji *cup* (Burrow and Emeneau 1984)

b. 'wing': Tamiḻ *ciṟai, iṟai, iṟakkai*, etc. (NEL [irʌkːʌi]), Malayāḷaṁ *iṟaku, ciṟaku* (NEL [t͡ʃiragɨ̈]), Kannaḍa *eṟake, ṟekke*, etc. (NEL [rɛkːe]), Telugu *eṟaka, ṟekka*, etc. (NEL [ɽekːʌ]) (Burrow and Emeneau 1984)

c. 'to revolve, turn around': Tamiḻ *curṟu* (NEL [t͡ɕutːu] 'wrap'), Malayāḷaṁ *curṟuka* (NEL [t͡ʃutːuga]), Kannaḍa *suttu* (NEL [sʊʈ̪ːu]), Telugu *cuṭṭu* (NEL [t͡ɕuʈːu] 'wrap') (Burrow and Emeneau 1984)

According to Andronov (2003, p. 74), initial /t͡ɕ/ became /s/ in Tamiḻ and sometimes in Kannaḍa, but not in Malayāḷaṁ and Telugu, and was lost in some words of all four languages. However, as I already mentioned in the previous section when discussing /s/ as a proto-sound, Andronov generally uses the symbol *s* in Tamiḻ words where you would normally expect *c*. According to Schiffman (1999, pp. 9f.) and Keane (2004, pp. 112ff.), /t͡ɕ/ is pronounced [s] intervocally in Tamiḻ, and its word-initial pronunciation is largely speaker-dependent. Since word-initial Tamiḻ *c* is consistently transcribed as [t͡ɕ] in NorthEuraLex, I will assume the sound change t͡ɕ > s / # _ for Kannaḍa only.

	Tamiḻ	**Malayāḷaṁ**	**Kannaḍa**	**Telugu**
***ʈ**	AN_{76f},$KR_{148,163ff}$			
-ɖ-	ɖ	ɖ	ɖ	ɖ
ɳɖ	ɳɖ	ɳɖ	ɳɖ	ɳɖ
ʈː			ʈ | Vː _	ʈ | Vː _
	ʈː	ʈː	ʈː	ʈː
ɳʈː	ʈː	ʈː	ɳʈ	ɳʈ
***k**	$AN_{47f,71ff}$,$KR_{128f,149f,163ff}$			
k-	t͡ɕ | _ [FRONT]	t͡ɕ | _ [FRONT]		t͡ɕ | _ [FRONT]
			g | _ V [SON.] (~)	g | _ V [SON.] (~)
	k	k	k	k
-g-	ɣ	g	g	g
ŋg	ŋg	ŋː	ŋg	ŋg
kː			k | Vː _	k | Vː _
	kː	kː	kː	kː
ŋkː	kː	kː	ŋk	ŋk

3.1.4.3 Sonorants

The sonorants are completely unchanged in Tamiḻ and Malayāḷaṁ. Kannaḍa has changed word-initial [ʋ] and entirely lost [ɻ], while Telugu systematically replaced all retroflex sonorants.

	Tamiḻ	**Malayāḷaṁ**	**Kannaḍa**	**Telugu**
***N**	AN_{82ff},$KR_{137ff,150f}$			
m	m	m	m	m
n̪	n̪	n̪	n̪	n̪
ɲ	ɲ	ɲ	n̪	n̪
ɳ	ɳ	ɳ	ɳ	n̪
ŋ	ŋ	ŋ	ŋ	ŋ
***ʋ**	$AN_{55f,85f}$,$KR_{141f,154}$			
			b | # _	
	ʋ	ʋ	ʋ	ʋ
***l**	AN_{85},KR_{153}			
	l	l	l	l
***ɭ**	AN_{87f},KR_{153f}			
	ɭ	ɭ	ɭ	l

***ɾ**	AN_{84f},KR_{151f}			
	ɾ	ɾ	r	r
	Tamiḻ	**Malayāḷaṁ**	**Kannaḍa**	**Telugu**
***ɻ**	AN_{86f},KR_{152f}			
	ɻ	ɻ	r | _ C ɭ	r | C _ ɖ
***j**	AN_{84},KR_{154}			
	j	j	j	j

3.1.5 Challenges

Tamiḻ, Malayāḷaṁ, Kannaḍa and Telugu are still noticeably similar and most of the sound changes are quite transparent. The original vowel system, usually the part of the phoneme inventory that is especially prone to sound changes, has been well conserved in all four languages. Having the extremely conservative Tamiḻ, "which in the Dravidian family retained [...] the greatest archaism and the hihgest [sic!] degree of propinquity to its Proto-Dravidian ancestor" (Andronov 2003, p. 24), and its sister language Malayāḷaṁ in the sample is another facilitation, since they have conserved unsteady phonemes such as alveolar /t/ and the retroflex approximant /ɻ/ that were lost in most of the other Dravidian languages (Krishnamurti 2003, p. 48). While the sample is restricted to South Dravidian languages, all proto-phonemes have been preserved in at least one of these languages, so it should be possible to satisfyingly reconstruct Proto-Dravidian from them.

A significant challenge, however, is posed by the Dravidian languages' close contact to their Indo-Aryan neighbors, in particular Sanskrit. While Tamiḻ is still rather resistant to loanwords, extensive borrowing has been going on in its relatives (Krishnamurti 2003). The percentage of Sanskrit material in the lexicons of Malayāḷaṁ, Kannaḍa and Telugu is believed to be well over 50 % (Staal 1963). Malayāḷaṁ is particularly inclined to borrowing, having a long literary tradition of blending Sanskrit and Dravidian material (Menon 1978). With the Sanskrit loanwords, Malayāḷaṁ has imported both voiced and voiceless aspirated stops as well as the fricatives /ɕ/, /ʂ/ and /h/ into its phonetic inventory (Asher and Kumari 1997). While not every speaker distinguishes all of these sounds, some do, so they must all be transcribed accordingly. A human linguist familiar with Sanskrit can easily identify and ignore these loanwords for reconstruction of Proto-Dravidian, but the overwhelming evidence for these Indo-Aryan sounds can pose quite a challenge for automated methods when the data comes without proper loanword annotation.

Even more challenging is the automated detection of genetic relationships. The huge amount of Sanskrit and nowadays also English lexical material could easily lead a system to place the Dravidian with the Indo-Aryan languages instead of assuming an independent language family. It might also distance the conservative Tamiḻ from its close relatives due to the seemingly small amount of shared lexical items and the differences in the phoneme inventories.

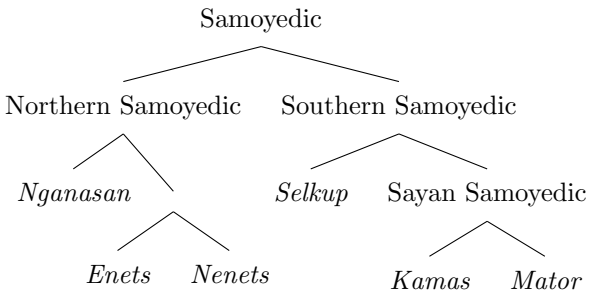

Figure 3.2: The traditional Samoyedic language group tree.

These problems hold especially for the NorthEuraLex database, which was designed to represent the lexicons of the modern languages and contain the words that are actually used nowadays, not so much to show off cognacy and genetic relationships in the languages' lexical inventories. As such, it naturally contains many loanwords for the Dravidian languages, but does not yet have loanword annotation. Since the concepts covered by the database have been selected with their stability in mind and are not so prone to borrowing, the share of loanwords among the NorthEuraLex entries for the Dravidian languages is probably much lower than in the overall lexicon of the modern Dravidian speaker. Still, any system for proto-reconstruction using the NorthEuraLex data must be able to discard the signals sent by the loanwords in some way.

3.2 Samoyedic

The Samoyedic languages form one of the major branches of the Uralic language family (Janhunen 1998; Sammallahti 1988; Hammarström, Forkel, and Haspelmath 2019). They are spoken in Northwest Siberia, mostly in the region between the Ob' and Yenisei rivers (Janhunen 1998; Hammarström, Forkel, and Haspelmath 2019). There are six known Samoyedic languages: Nganasan, Enets and Nenets, traditionally grouped together as Northern Samoyedic, and Selkup, Kamas and Mator, traditionally grouped together as Southern Samoyedic (Janhunen 1998; Hajdú 1988). This division, however, is not uncontroversial. An alternative taxonomy has Nganasan and Mator split off early and groups Enets together with Nenets and Selkup together with Kamas (Janhunen 1998; Hammarström, Forkel, and Haspelmath 2019). Figure 3.2 shows the language tree that results from the traditional grouping.

With the exception of Nenets, the Samoyedic languages are poorly documented and endangered by extinction or, in the case of Kamas and Mator, have already become extinct (Janhunen 1998). Nenets is still surviving with over 20,000 speakers (Janhunen 1998; Lewis, Simons, and Fennig 2015). The NorthEuraLex database only contains dialects of the four living Samoyedic languages, namely Nganasan, Enets (Forest dialect), Nenets (Tundra dialect) and Selkup (Northern dialect). The following sections will therefore only be concerned with the phonetic evolution of these four languages.

The Uralicist literature in general does not use the IPA for phonetic transcription but the

	Front		Central		Back	
High	*i* i	*ü* y			*į* ɯ	*u* u
Mid	*e* e	*ö* ø	*ə̂* ə		*ę* ɤ	*o* o
Low	*ä* æ					*å* ɒ

Table 3.3: The reconstructed vowel inventory of Proto-Samoyedic (UPA in italics to the left, IPA to the right).

Uralic Phonetic Alphabet (UPA), a transcription scheme first suggested by Setälä (1901) which is particularly suited for the phonetic systems of the Uralic languages. Since the NorthEuraLex database uses IPA transcription, I needed to find suitable IPA equivalents for the UPA symbols discussed in the Uralicist sources. The following two sections will therefore not only discuss the Proto-Samoyedic sound inventories, but also justify my choices for the IPA transcription.

3.2.1 Proto Vowels

Up until the end of the 20th century, there was general agreement that the Proto-Samoyedic sound inventory contained eleven vowels, as arranged in Table 3.3 (Sammallahti 1988; Janhunen 1998; Mikola 2004). However, especially the slightly unclear evolution of the 'reduced' Proto-Samoyedic vowel *ə̂* has led to proposals of additional proto-vowels. In 1993, Eugen Helimski proposed a second 'reduced' vowel on the basis of Nganasan data. In 2005, he published the paper "The 13th Proto-Samoyedic vowel" where he not only extended the inventory by yet another vowel, but also reinterpreted the phonetic values of the previously accepted proto-vowels. Recently, there have even been suggestions for a 14th and 15th Proto-Samoyedic vowel (Normanskaja 2018). Janhunen's (1977) etymological dictionary and all of the comprehensive works about Samoyedic sound change, however, are from before these additions and are thus still based on the old vowel system. Mikola (2004) already acknowledges Helimski's 12th vowel, but does not incorporate it in any sound changes discussed. Therefore, I have no choice but to still assume the traditional Proto-Samoyedic inventory of eleven vowels as displayed in Table 3.3.

The phonetic value of the UPA symbols *e*, *i*, *o*, *u*, *å*, *ä*, *ö* and *ü* is rather clear from Setälä's (1901) description:

- *ä* is described as "a 'broad *ä*', as in Swedish before **r**" (orig. "ein 'breites *ä*', wie im schwedischen vor **r**"; p. 36), which corresponds to IPA [æ].
- *å* is described as a "labialized **a**" (orig. "labialisiertes **a**"; p. 38), which corresponds to IPA [ɒ].
- *ü* is described as "*ü* as in German" (orig. "*ü* wie im deutschen"; p. 38), which corresponds to IPA [y].
- Setälä notes "that with *e i o ö u ü* [...], one denotes rather open ('wide') variants of the respective sounds"[1] (p. 36), which suggests that the corresponding IPA transcriptions

[1] my translation, orig. "dass man mit *e i o ö u ü*, wenn nicht das gegenteil hervorgehoben ist, ziemlich offene ('wide') varianten der betreffenden laute bezeichnet"

would be those of the tense vowels [e], [i], [o], [ø], [u] and [y].

The diacritic ◌̯ (breve below) "denotes the retraction of the tongue next to the retraction of the lips, e.g. *a*, *e̯* (= est. **õ**), *i̯* (= a sound that is close to russ. ы, although without being diphthongish)"[2] (Setälä 1901, p. 40). In a vowel chart on p. 41, *e̯* and *i̯* are marked as vowels with little opening of the mouth (like *o* and *u*), retracted lips (as opposed to protruded lips with rounding) and retracted tongue (as opposed to *e* and *i* with protruded tongue). They are thus the back counterparts of the high unrounded front vowels [e] and [i] and can therefore be transcribed in IPA as [ɤ] and [ɯ], respectively. This also matches Estonian *õ* [ɤ] and is close to Russian ъи [ɨ].

Translating *ə̑* into IPA is a bit of a challenge, also since the symbol used for this sound is not consistent among Uralicists. Janhunen (1977), Janurik (1982) and Mikola (2004) use *ə̑*, while Mikola (1988) uses *ŏ*, Sammallahti (1988) uses *ɵ* and Janhunen (1998) uses *ø*. I stick with the majority here and display it as *ə̑*. This is also the only one of the four symbols that is mentioned by Setälä (1901).

Setälä (1901) states that "[b]y inversion of the vowel signs, one creates signs for 'indifferent' or otherwise incomplete vowels"[3] (p. 39). This indicates that *ə̑* is a lax version of e̯ ([ɤ]), for which there is no symbol in IPA. Janhunen (1977), Sammallahti (1988), Mikola (1988) and Janhunen (1998) display the sound in the horizontal center of their vowel charts, indicating that it is in between front and back vowels, while Mikola (2004) places it in the same column as the rounded front vowels *ü* [y] and *ö* [ø]. Neither of them gives a vertical placement of *ə̑*, instead they list it as the 'reduced' vowel below the regular vowels. Janhunen (1998) describes it as "being quantitatively shorter and prosodically weaker" (p. 463). No more concrete specification of its phonetic quality is given. Since it appears to be generally lax or "weak" vowel of underspecified place of articulation, I have chosen to transcribe it as the schwa [ə].

It is assumed that Proto-Samoyedic had eight falling diphthongs, [iə̯], [yə̯], [eə̯], [øə̯], [uə̯], [ɤə̯], [oə̯] and [ɒə̯], with the second element [ə̯] deriving from an mostly undefined (but probably velar) sound 'x' reconstructed for Proto-Uralic (Mikola 2004, p. 22f; Sammallahti 1988, p. 482, 485).

3.2.2 Proto Consonants

There is wide agreement among Uralicists that the Proto-Samoyedic phoneme inventory consisted of 13 consonants, namely the ones displayed in Table 3.4 (Sammallahti 1988; Mikola 2004; Janhunen 1998). Sammallahti (1988) mentions a "possible secondary **ś*", but puts it in brackets in his consonant table. Mikola (2004, p. 29ff) also considers **ś*, but notes that it can only be derived from Proto-Uralic **ć*, which is not reconstructed by many scholars. Janhunen (1998) does not discuss **ś* at all. I have therefore decided not to include this phoneme in my gold standard.

[2] my translation, orig. "bezeichnet die zurückziehung der zunge nebst zurückziehung der lippen, z. b. *a*, *e̯* (= est. **õ**), *i̯* (= ein laut, zu dem russ. ы nahe steht ohne jedoch diphthongisch der [sic!] sein)"

[3] my translation, orig. "Durch umkehrung der vokalzeichen stellt man zeichen für 'indifferente' oder in einer oder anderer hinsicht unvollkommene vokale her"

	Labial	Alveolar	Retroflex	Palatal	Velar
Stop	*p* p	*t* t			*k* k
Affricate			*c* t͡s̻		
Nasal	*m* m	*n* n		*ń* ɲ	*ŋ* ŋ
Fricative		*s* s			
Lateral		*l* l			
Trill		*r* r			
Approximant	*w* w			*j* j	

Table 3.4: The reconstructed consonant inventory of Proto-Samoyedic (UPA in italics to the left, IPA to the right).

The UPA transcriptions of the Proto-Samoyedic consonants are for the most part the same as their IPA equivalents. The acute accent on consonants marks palatalization (Setälä 1901, p. 40), hence *ń* stands for either [nʲ] or [ɲ]. For the sake of simplicity, I choose to transcribe it as the latter.

The phonetic quality of *c* is more difficult to determine. In all of the consonant charts for Proto-Samoyedic (Janhunen 1977; Sammallahti 1988; Mikola 2004), *c* is in the same column as *t*, *n*, *s*, *l* and *r*, but receives its own row, suggesting that it is alveolar or at least coronal but that its manner of articulation differs from the other alveolar/coronal sounds (in particular the plosive). Considering the usual conventions for the phonetic values of the letter *c*, it is highly probable that it denotes an alveolar affricate, for instance [t͡s] or [t͡ʃ]. However, Setälä (1901) does not list it; in fact, he explicitly rejects the idea of having single symbols for sequences of multiple sounds, like affricates (p. 34). According to Sammallahti (1988), in Proto-Uralic, "/c/ was retroflex (cacuminal)" (p. 482). This is in accordance with Janhunen (1998), who describes Proto-Samoyedic *c* as "retroflex and/or affricated" (p. 462). I therefore transcribe it as [t͡s̻].

3.2.3 Phonotactics

The majority of Proto-Samoyedic reconstructions in the etymological dictionary of Janhunen (1977) are mono- or bisyllabic words, but some non-derived forms have up to three syllables (Janurik 1982, p. 42f). Only single consonants are allowed word-initially and word-finally, but clusters of two consonants can occur between vowels. Most of these are combinations of nasal and obstruent, but sequences of two plosives or plosive and fricative may also occur. There are five instances of three-consonant clusters in Janhunen (1977), but all of these begin with the glide /j/ (Janurik 1982, p. 42ff).

The full vowel inventory can only be observed in first syllables. Non-initial syllables show a reduced set of vowels, whose exact composition is still a matter of dispute (Mikola 2004, p. 23ff). The popular opinion also presented by Sammallahti (1988) and Janhunen (1998) is that it consisted of the three vowels [ɒ], [æ] and [ə]. All consonants could generally appear in all possible positions, with the exception of [r] and [ŋ], which never appear

word-initially (Sammallahti 1988, p. 482), and [ŋ], which is never found in the coda[4] (Mikola 2004, p. 50).

3.2.4 Sound Changes

In this section, I present the sound changes from Proto-Samoyedic into Nganasan, (Forest) Enets, (Tundra) Nenets and (Northern) Selkup. $^{(\sim)}$ indicates an irregular sound change. The first row of each table refers to the sources for the individual languages, with the respective page numbers in subscript.

The main source is the posthumously published and edited academic dissertation of Tibor Mikola (2004), abbreviated MI, an extensive overview over the phonological and morphological history of the Samoyedic languages. Janurik (1982) (JK) provides a complete list of all sound correspondences that can be extracted from the etymological dictionary of Janhunen (1977) with their frequencies. Unfortunately, he only published the consonant correspondences. As an additional source for the vowels (and for some consonants), I use Sammallahti (1988) (SA), who gives tabular overviews over both vowel and consonant correspondences. However, he gives no information on conditioned sound changes for vowels and only little for consonants, and lists only word-initial consonant changes. Finally, Janhunen (1998) (JN) discusses a few Samoyedic sound changes, but does not provide an exhaustive list.

3.2.4.1 Vowels

The Proto-Samoyedic vowel inventory has been reduced greatly in Nganasan, Enets and Nenets, where the front-back contrasts [i∼ɯ], [y∼u], [e∼ɤ], [ø∼o] and [æ∼ɒ] have been eliminated. In Nenets, the lost front-back opposition is reflected in the palatalization of consonants preceding formerly front vowels. Selkup is the most conservative of the four languages, having retained most of the Proto-Samoyedic vowels (most notably [y] and [ɤ]). Nganasan is the only Samoyedic language that still has the reduced vowel [ə] (Mikola 2004, p. 72f).

	Nganasan	**Enets**	**Nenets**	**Selkup**
***æ**	SA_{495},JN_{467},MI_{73}	SA_{495},MI_{63}	SA_{495},MI_{39}	SA_{495},MI_{83}
	ɑ	e	ɑː	ɑ
***ɒ**	SA_{495},JN_{467},MI_{78}	SA_{495},MI_{64}	SA_{495},MI_{39}	SA_{495},MI_{83}
	o | # (C) _ u	ɑ	ɑ	ɑ

[4] Mikola (2004) notes, however: "With respect to the consonant *-ń*, a PS [Proto-Samoyedic] etymology with the sound *-ń* can perhaps just not be shown due to its low frequency. I, in any case, don't see a principal reason for the assumption that *-ń* could not have occurred word-finally or in pre-consonantal position." (p. 50; my translation, orig. "Was den Konsonanten *-ń* betrifft, so kann eine PS Etymologie mit dem Laut *-ń* vielleicht nur wegen dessen niedriger Frequenz nicht nachgewiesen werden. Ich jedenfalls sehe keinen prinzipiellen Grund für die Annahme, dass *-ń* nicht im Wortauslaut bzw. in präkonsonantaler Position hätte vorkommen können.")

***e**	SA_{495},MI_{74ff}	SA_{495},MI_{63}	SA_{495},MI_{39}	SA_{495}
	ɑ (∼)			
	e	e	e	e
	Nganasan	**Enets**	**Nenets**	**Selkup**
***ø**	SA_{495},JN_{467},MI_{73}	SA_{495},MI_{62}	SA_{495},MI_{39}	SA_{495}
	u	o	oː	y (⋆)

(⋆) Mikola (2004) claims, without providing any examples, that "PS **ö* became *ü*" (p. 84) in Selkup. There are only four entries in Janhunen (1977) that contain **ö*, and only one of those has a reflex that I could find in Irikov (1988), the source dictionary for the NorthEuraLex Northern Selkup entries, namely **nöǰnå* "burbot" ("orig. "Quappe"), which is listed as *нюни* [nyni] under ru. *налим*. This indicates that the sound change ø > y proposed by Sammallahti (1988) is correct for the NorthEuraLex source, even though the only affected entry does not occur in the actual word list, rendering this sound change impossible to detect for SoInEn.

	Nganasan	**Enets**	**Nenets**	**Selkup**
***i**	SA_{495},$MI_{73,76f}$	SA_{495}	SA_{495},MI_{39}	SA_{495}
	ɯ | [LABIAL] _			
	i	i	i	i
***y**	SA_{495},JN_{467},MI_{73}	SA_{495},MI_{62}	SA_{495},MI_{39}	SA_{495}
	i	u	u	y
***ɤ**	SA_{495},MI_{74}	SA_{495},MI_{63}	SA_{495},MI_{39}	SA_{495}
	ɯ | _ r	u | [LABIAL] _		
	ɑ	i	eː	ɤ
***o**	SA_{495},JN_{467},MI_{73}	SA_{495},MI_{64}	SA_{495},MI_{39}	SA_{495}
	u	o	o	o
***ɯ**	SA_{495},JN_{467},MI_{73}	SA_{495},MI_{63}	SA_{495},MI_{39}	SA_{495}
		u | [LABIAL] _		
	i	i	i	i
***u**	SA_{495}	SA_{495}	SA_{495}	SA_{495}
	u	u	u	u
***ə**	MI_{73}	SA_{495},MI_{62}	SA_{495},MI_{38}	SA_{495},MI_{83}
	e | [PALATAL] _			
	ə	o	ɑ	ɑ

***∅**	SA_{497},MI_{81}	SA_{497},MI_{54f}
	^{j} | t,n,s,l _ i,y,e,ø	^{j} | C _ [FRONT]

None of the Proto-Samoyedic diphthongs have been retained in their reconstructed form in any of their descendant languages, making them especially hard to infer for any automated method, and probably impossible for SoInEn in its current form. In Nganasan and Enets, they were mostly collapsed into a smaller set of diphthongs, while they introduced long vowels to the Nenets vocalic system. Selkup, again, is rather conservative and mostly reduces them to monophthongs according to the prominent vowel.

	Nganasan	**Enets**	**Nenets**	**Selkup**
***VV**	SA_{495},MI_{77}	SA_{495},MI_{63}	SA_{495},MI_{40}	SA_{495}
ɒə̯	uɑ	ɑː	ɑː	u
eə̯	ie	e	eː	e (⋆)
øə̯	uɑ	iɑ	oː	æ
iə̯	ie	io	iː	i
yə̯	i	io	uː	y
oə̯	uɑ	uɑ	oː	o
ɤə̯	ɑ	iɑ	eː	ɑ
uə̯	uɑ	e	uː	u

(⋆) Sammallahti (1988) does not list the diphthong */eə̯/ and Mikola (2004) does not discuss the reflexes of the Proto-Samoyedic diphthongs in Selkup at all. In Janhunen (1977), the Selkup reflex of **keə̂j* 'tongue' is *še* or *se*, indicating that there might have been a sound change eə̯ > e in Selkup.

3.2.4.2 Obstruents

The environments for different consonant change in the Samoyedic languages are usually word-initially, intervocally, and in the syllable coda. For the sake of brevity, I refer to these general contexts in the first column as **C-** (unless otherwise specified, read: C | # _), **-C-** (C | V _ V) and **-C** (C | _ C, _ #).

The Proto-Samoyedic obstruents usually (but not always) each undergo the same basic changes: In Enets and Nenets, plosives are voiced intervocally. In the coda, obstruents are reduced to a glottal stop in Nganasan, Enets and Nenets. In Enets, they are even deleted entirely before voiceless plosives (abbreviated P). Selkup again proves to be very conservative in this respect.

All four languages palatalize [k] before front vowels, albeit to varying degrees. Selkup additionally changes [k] to [q] before non-high vowels. [t͡s] is merged with [t] in all four languages.

	Nganasan	**Enets**	**Nenets**	**Selkup**
***p**	JK_{70f},MI_{82}	JK_{70f},$MI_{42ff,65ff}$	JK_{70f},MI_{55}	JK_{70f}
p-	h	p $^{(\star)}$	p	p
-p-	h	b	b	p
-p		∅ | _ P		
	b	ʔ	p	p

$^{(\star)}$ Sammallahti (1988, p. 497) lists a word-initial sound change p > f for Enets (there called 'Yenissei'). However, as Mikola (2004, p. 68) notes, this only applies to some Enets dialects. Since word-inital */p/ has been retained in the NorthEuraLex data for Enets (cf. e.g. *пя* [pʲa] 'tree' < **pä*), I do not assume this sound change here.

	Nganasan	**Enets**	**Nenets**	**Selkup**
***t**	JK_{74ff},MI_{80}	JK_{74ff},MI_{65ff}	JK_{74ff},$MI_{42ff,55}$	JK_{74ff},SA_{498}
t-	t	t	t	t
-t-	t	d	d	t
-t		∅ | _ P		
	ʔ	ʔ	ʔ	t
***s**	JK_{73},MI_{80}	JK_{73},MI_{65}	JK_{73},MI_{42ff}	JK_{73}
s-	s	s	s	s
-s-	s	s	s	s
-s		∅ | _ P		
	ʔ	ʔ	ʔ	s
***t͡s**	JK_{57f},SA_{497},$MI_{80,82}$	JK_{57f},SA_{497},MI_{65}	JK_{57f},SA_{497},$MI_{42ff,55}$	JK_{57f},SA_{497},MI_{85}
t͡s-	sʲ | _ i,e			ʃ | _ i,e
	t	t	t	t
-t͡s-	sʲ | _ i,e			ʃ | _ i,e
	t	d	d	t
-t͡s		∅ | _ P		
	ʔ	ʔ	ʔ	t
***k**	JK_{60ff},SA_{497},MI_{80f}	JK_{60ff},SA_{497},MI_{65}	JK_{60ff},SA_{497},$MI_{42ff,55}$	JK_{60ff},SA_{497},$MI_{85,86}$
k-	sʲ | _ i,e	s | _ [FRONT]	sʲ | _ [FRONT]	ʃ | _ i,e q | _ [-HIGH]
	k	k	χ	k
-k-	sʲ | _ i,e	s | _ [FRONT]	sʲ | _ [FRONT]	ʃ | _ i,e q | _ [-HIGH]
	k	h	χ	k
-k	k | _ P	∅ | _ P		
	ʔ	ʔ	ʔ	k

3.2.4.3 Nasals

The Proto-Samoyedic nasals are generally more stable than the obstruents, though they are also partially reduced to glottal stops in Enets and Nenets (but not in Nganasan). Selkup preserves them completely.

	Nganasan	**Enets**	**Nenets**	**Selkup**
***m**	JK$_{65}$,MI$_{82}$	JK$_{65}$,MI$_{66f}$	JK$_{65}$,MI$_{56}$	JK$_{65}$,SA$_{498}$
m-	m	m	m	m
-m-	m	∅	w	m
-m	∅ | _ # $^{(\star)}$			
	m	ʔ	m	m
***n**	JK$_{66f}$,MI$_{82}$	JK$_{66f}$,MI$_{66}$	JK$_{66f}$,MI$_{42ff}$	JK$_{66f}$,SA$_{498}$
n-	n	n	n	n
-n-	n	n	n	n
-n	∅ | _ # $^{(\star)}$			
	ŋ | _ #			
	n	ʔ	ʔ	n
***ɲ**	JK$_{68}$	JK$_{68}$	JK$_{68}$	JK$_{68}$
ɲ-	ɲ	ɲ	ɲ	ɲ
-ɲ-	∅	∅	j	ɲ
***ŋ**	JK$_{69}$,MI$_{82}$	JK$_{69}$,MI$_{66f}$	JK$_{69}$,MI$_{42ff}$	JK$_{69}$
-ŋ-	ŋ	∅	ŋ	ŋ
-ŋ	∅ | _ # $^{(\star)}$			
	ŋ	ʔ	ʔ	ŋ

$^{(\star)}$ Word-final nasal deletion in Nganasan only applies in words with more than one syllable (Mikola 2004, p. 82).

Nasal-obstruent clusters deserve special mention particularly because of Enets, which reduces them to geminate voiced stops matching the place of articulation of the cluster's obstruent.

	Nganasan	**Enets**	**Nenets**	**Selkup**
***NC**	JK$_{82ff}$	JK$_{82ff}$,MI$_{66}$	JK$_{82ff}$,MI$_{56}$	JK$_{82ff}$
mp	mb	bː	mb	mp
mt	mt	dː	md	mt
nt	nt	dː	n	nt
nt͡s̻	nt	dː	n	nt
ŋt	t	dː	ŋd	ŋt
ŋk	ŋk	gː	ŋg	ŋk

In Nganasan and Nenets, a prothetic nasal ([ŋ] or [ɲ], depending on the following vowel) has been inserted in words starting with a vowel.

	Nganasan	**Enets**	**Nenets**	**Selkup**
***∅**	JK$_{56}$,SA$_{497}$,MI$_{81}$		JK$_{56}$,SA$_{497}$,MI$_{54}$	
	ɲ | # _ i,e		ɲ | # _ [FRONT]	
	ŋ | # _ V		ŋ | # _ V	

3.2.4.4 Liquids and Glides

Liquids are unchanged in all languages except Enets, where non-initial [l] has been merged with [r] and both are reduced to a glottal stop in the coda.

	Nganasan	**Enets**	**Nenets**	**Selkup**
***l**	JK$_{63f}$	JK$_{63f}$,MI$_{66,69}$	JK$_{63f}$	JK$_{63f}$
l-	l	l	l	l
-l-	l	r	l	l
-l	l	ʔ	l	l
***r**	JK$_{71f}$	JK$_{71f}$,MI$_{66}$	JK$_{71f}$	JK$_{71f}$
-r-	r	r	r	r
-r	r	ʔ	r	r

The glides [w] and [j] are very unstable in the Samoyedic languages, having been fortified into plosives in Nganasan, Enets and, most prominently, Selkup.

	Nganasan	**Enets**	**Nenets**	**Selkup**
***w**	JK$_{77f}$,SA$_{497}$,MI$_{80}$	JK$_{77f}$,SA$_{497}$,MI$_{67f}$	JK$_{77f}$,SA$_{497}$,MI$_{54,56}$	JK$_{77f}$,SA$_{497}$,MI$_{85}$
w-			j | _ [FRONT]	
	b	b	w	k
-w-	b	∅	b	∅
***j**	JK$_{58ff}$,MI$_{80f}$	JK$_{58ff}$,MI$_{68,69}$	JK$_{58ff}$,MI$_{40}$	JK$_{58ff}$,SA$_{497}$,MI$_{85}$
j-	dʲ	dʲ	j	t͡ʃ
-j-	dʲ	j	j	d͡ʒ
-j	j	∅	ː	j

3.2.5 Challenges

While the Dravidian languages are still quite similar to each other, the Samoyedic languages have often diverged rather drastically, making them a challenging test case for computational historical linguistics. As Janhunen (1998) notes:

> "The phonological correspondences among the Samoyedic languages are for the most part transparent to the professional eye, although a naïve native speaker

> of any single Samoyedic language would often find it impossible to recognize cognate items even in the immediately neighbouring language." (p. 466)

Indeed, most of the consonant changes, such as palatalization of [k], glottalization of obstruents or the spirantization of [p] into [h] (cf. Kannaḍa), observed in the Samoyedic languages are rather common and not too surprising, and the vowels have perceptually very similar reflexes in all of the daughter languages. What makes Samoyedic a complicated case for naïve natives and especially for naïve computers is 1) the pervasiveness of sound change, in that in many cases, the original sound is preserved in only a single or even none of the descendants, 2) the scarcity of lexical material in general, and 3) the strong dialectal variation in modern Samoyedic languages.

Remember the Nganasan, Nenets and Selkup cognates for "ten" discussed in section 1.3.1 (*bî'* [biːʔ], *ju'* [juʔ] and *köt* [køt], respectively): Deciding that these three words have a common ancestor (**wüt*) is not a trivial task for a computer that relies heavily on symbol and sound similarity, even though the sound changes they have gone through are quite regular. Since both vowels and consonants where heavily affected by sound change in Samoyedic (as opposed to Dravidian, where there are basically no vowel changes at all), every single phoneme in a word can be affected by sound change, as in the "ten" example.

This does not only affect the quality of automated cognate judgments, but especially that of reconstruction, a task that already involves a lot of speculation for human linguists when none of the reflexes can be considered proto-sound candidates, as in the correspondences b/j/k and iː/u/ø. This is especially true of proto-reconstructions that are not attested in any of the daughter languages, as with Proto-Samoyedic *[ɯ], which has become [i] in all four descendants, and can be distinguished from *[i] only in sound correspondences occurring in a very specific context, namely after labial consonants, where we find both i/u/i/i (going back to *[ɯ]) and ɯ/i/i/i (going back to *[i]). Reasoning that these two extremely rare correspondences justify the reconstruction of two separate proto-sounds that both yield the correspondence i/i/i/i outside of this context, all while not over-interpreting equally rare nonsensical correspondences in other places, is an immensely difficult (and likely impossible) task for a computer.

This problem is amplified particularly in Samoyedic by the noisiness of the lexical material available. As initially mentioned, the Samoyedic languages (except Tundra Nenets) are rather poorly documented, and the few sources available are likely documenting different dialects, often leading to considerable variation in the lexical entries. In Janhunen (1977), different sources give the Nganasan cognates of **mə̑kå* "back" as *мáку*, *móku* or *məku*, those for **jimä* "glue" as *jimi* or *d'iḿi*, and those for **wåt^{3}wə̑* "bedding" as *bóba* or *bəbə*. Most of these inconsistencies concern the vowels, which are especially difficult to reconstruct anyway given their great variation. Obviously, the sound changes that have taken place according to the sources differ, which can be due to their respective age, but also due to different dialects observed. The Enets cognates for **wåt^{3}wə̑*, for example, are given as *bá'a* or *bāa*, and similarly, those for **sejt^{3}wə̑* "seven" are *se'o* or *sɛo*. In one of these dialects, the glottal stops that earlier replaced coda obstruents have vanished. This is of course particularly problematic when mixing sources, but is also an issue with respect to

the above given gold standard of sound changes when relying on a single source: The one for the Enets data in NorthEuraLex, for instance, has no glottal stops represented at all, making it impossible to reconstruct the corresponding gold standard sound changes on its basis.

Finally, internal phonological processes of the individual languages can distort the results. Nganasan words, for instance, undergo three different types of consonant gradation (Mikola 2004, p. 79f): The first operates depending on the number of preceding morae, a second, slightly different type then applies to those left untouched by the first type, and finally, nasals deleted by the first type are reintroduced in case the affected syllable begins with a nasal. Due to the extreme complexity of these phenomena, I have not incorporated their effects in the sound changes listed above, but they will of course be found in the lexical data, providing another source of noise.

4 SoInEn, a PSL Model for Sound Law Inference

This chapter introduces the Sound Law Inference Engine (SoInEn), a PSL model designed to infer proto-phoneme reconstructions for sound correspondences and conditioned sound laws from lexical data of modern languages. SoInEn's rules are modeled after the reasoning performed by human historical linguists when applying the comparative method. This could not only provide interesting insights into how much each of these reasoning steps contribute to the final outcome, it also allows the system to inform about its reasons for coming to a certain conclusion in a way that makes sense to the linguist user.

SoInEn is part of the larger EtInEn system for etymological inference. Those features of EtInEn that are relevant to SoInEn and the graphical user interface of SoInEn inside EtInEn are briefly introduced in section 4.1. Not all steps of the comparative method are feasible to be performed via PSL. Those tasks that had to be outsourced, notably cognacy judgment and alignment, as well as general linguistic knowledge submitted as observations are discussed in section 4.2. The actual inference of SoInEn had to be split into three separate inference phases for performance reasons, namely proto-inventory reconstruction, context detection and sound law inference. The predicates and rules of each phase as well as the procedure for generating candidate target atoms are discussed in sections 4.3, 4.4 and 4.5, respectively.

4.1 Integration into EtInEn

SoInEn was developed as a module of the under-development Etymological Inference Engine (EtInEn), a mainly PSL-based system solving various tasks in historical linguistics, such as cognacy and loanword detection, morphological analysis, sound law inference and proto-language reconstruction. In contrast to other computational tools in historical linguistics, where the role of the linguist user is usually limited to the provider of input and receiver of output, EtInEn is designed to be guided interactively by the user towards its result, accepting corrections and suggestions, and adapting its output accordingly. Due to the PSL backend, it is also able to provide human-readable explanations for its decisions, giving the user insights into its reasoning process. Any lexical database in CLDF

format can be input to EtInEn, allowing linguists to explore and analyze their own data sets.

Apart from developing the linguistic components, the EtInEn group has also built a wrapper around PSL, providing extended support for manipulating the atom database, managing multiple inferences on (partially) shared atoms, as well as interpreting and verbalizing inference results. While this infrastructure was primarily developed for EtInEn, it can also be used independently for other projects.

Finally, EtInEn comes with a graphical user interface for managing the input data, configuring and running the inferences, and inspecting the results for each linguistic component in a user-friendly, non-technical manner. In addition, it provides a component for inspection of all atoms with human-readable explanations for how the individual ground rules influenced the atom's belief value. This component, the *Fact Viewer*, is also going to be available as a standalone tool for use in other PSL projects.

Since SoInEn is embedded inside EtInEn, it is necessary to introduce some of EtInEn's features in more detail. First, there are specific naming conventions for predicates of EtInEn components, which I briefly introduce in section 4.1.1. EtInEn's PSL infrastructure communicates directly with the SQL database set up by LINQS PSL for atom storage, providing important new manipulation options for atoms, some of which are relevant for SoInEn (section 4.1.2). Finally, I present the EtInEn user interface, in particular the parts visualizing SoInEn, in section 4.1.3.

Links to the repositories containing the source code of SoInEn, EtInEn, and the PSL wrapper with the Fact Viewer can be found in appendix A.

4.1.1 Predicate Naming Conventions

The predicates of EtInEn PSL models follow strict naming conventions. The names of regular, "meaningful" predicates consist of exactly four characters, one uppercase followed by three lowercase. The initial uppercase character is a prefix signaling the linguistic domain of the predicate, whereas the three lowercase letters are its actual name. The name of the SoInEn predicate representing sound laws, for instance, is not `SoundLaw`, but `Plaw`, where `P` marks the 'phonology' domain, and `law` is the predicate's name. Other domain prefixes currently used inside EtInEn are, for example, `M` for 'morphology' or `E` for 'etymology'. SoInEn exclusively uses the `P` prefix.

There is an additional set of prefixes for auxiliary predicates that refer to those regular predicates. These auxiliary prefixes are attached before the domain prefix. Existential predicates, for instance, are marked by `X`. Hence, the name of the existential predicate of `Plaw` is `XPlaw`. Similarly, predicates that provide priors for regular predicates are prefixed `V` (for 'value').

4.1.2 Database Manipulation

The LINQS implementation of PSL stores a model's atoms in an SQL database. While the LINQS interface does provide some methods for inserting, retrieving and deleting

atoms, several functionalities required by EtInEn were missing, which is why its PSL infrastructure interacts directly with the SQL backend for atom manipulation.

One of the key features missing in the LINQS implementation was the possibility to *fixate* or *release* individual or sets of atoms, i.e. changing them to observations or targets after insertion[1]. Since EtInEn is supposed to be an interactive system, it is crucial that the user is able to select some atoms from an inference result for re-inference while keeping the belief values of the others fixed. Also, some EtInEn components, notably SoInEn, run several consecutive inferences with different targets, which at each step requires fixation of the previous inference's targets.

4.1.3 User Interface

The graphical user interface of EtInEn is developed within the JavaFX framework[2]. It consists of multiple windows, each hosting a single component or function of the system. This allows the user to freely arrange the individual parts of the system on his desktop, and facilitates working with several monitors. EtInEn's windows are not independent of each other, however: User interaction in one window is propagated through the entire system, allowing other windows to react as well. Figure 4.1 illustrates this: The Concepts window lists all semantic concepts represented in the provided database. Selecting the concept 'louse' filters the lexical data in the Forms window to only show translations for 'louse'. Finally, upon selecting the Swedish entry *lus* from the 'Forms' window, the Cogset window immediately displays the alignment of *lus* and its cognates.

In addition to the database inspection windows shown in Figure 4.1, there is a window for each inference component. The one I designed for SoInEn consists of three sections, only one of which can be expanded at a time: Initially, the Inference section is displayed, which can be seen in Figure 4.2. Here, the inference can be configured and started, and the user is informed about its progress while it is running. The input languages are selected in the Languages window.

Once the inference has finished, the second section can be opened to review the inferred proto-phoneme Inventory. As Figure 4.3 shows, the candidate consonants and vowels are arranged as in the IPA chart for convenient inspection. The color of the individual phonemes indicates their belief value, varying between white (0.0) and dark yellow (1.0), which can also be made explicit by hovering over the symbol.

The final Sound Changes section, shown in Figure 4.4, displays the sound correspondences found, their proto-reconstructions, the regular contexts in which they occur, and the sound laws deduced. Again, the color indicates belief, which is also displayed explicitly in the Value columns. The dropdown cells in the Support column list all cognate sets whose

[1]LINQS handles the distinction between observations and targets by separating the atom space in the database into several *partitions*. Those atoms that reside in *write partitions* are considered targets and will have their belief values changed during inference, while those in *read partitions* are considered observations. The `RDBMSDatabase` class does have a `moveToWritePartition` method, but no `moveToReadPartition`, which is actually the more common use case in EtInEn.

[2]Note that the EtInEn GUI is still under active development. While all of the features that I discuss are already fully functional, some windows contain placeholders for not-yet-implemented functions.

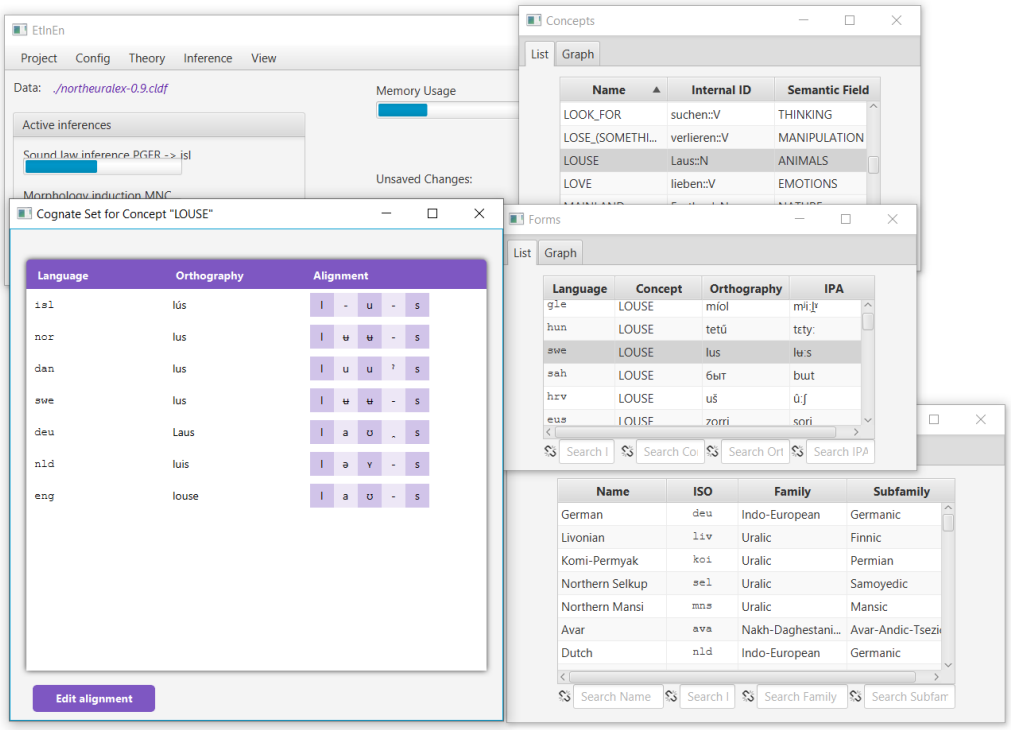

Figure 4.1: The main windows of the EtInEn user interface for inspecting the input database are informed about each other's state. Because the Swedish word for 'louse' was selected in the Forms window, the Cogset window displays the associated cognate set.

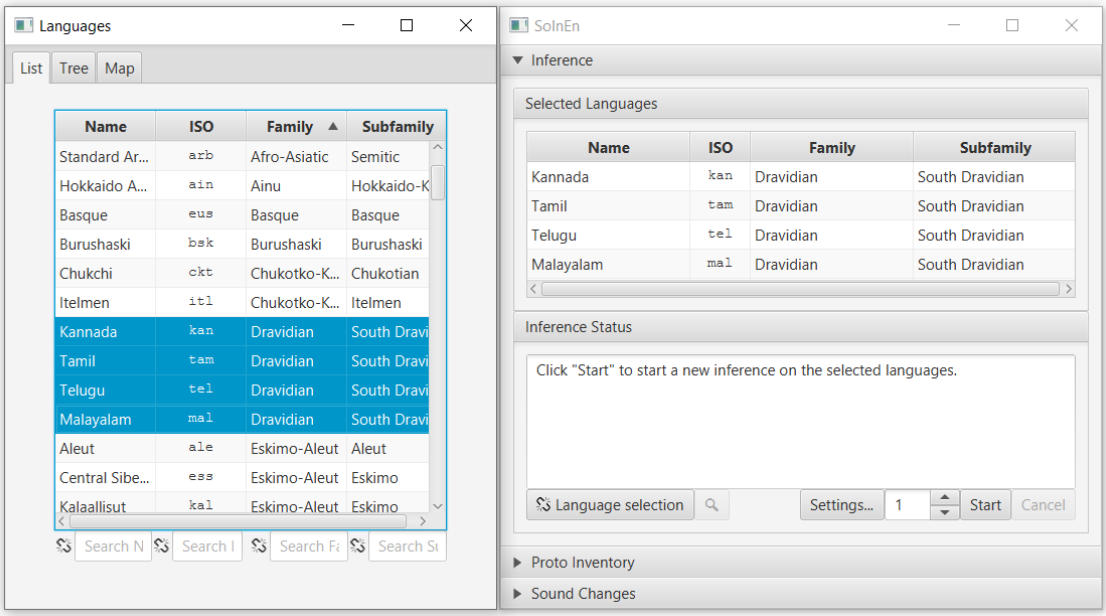

Figure 4.2: SoInEn's Inference pane is connected to the Languages window.

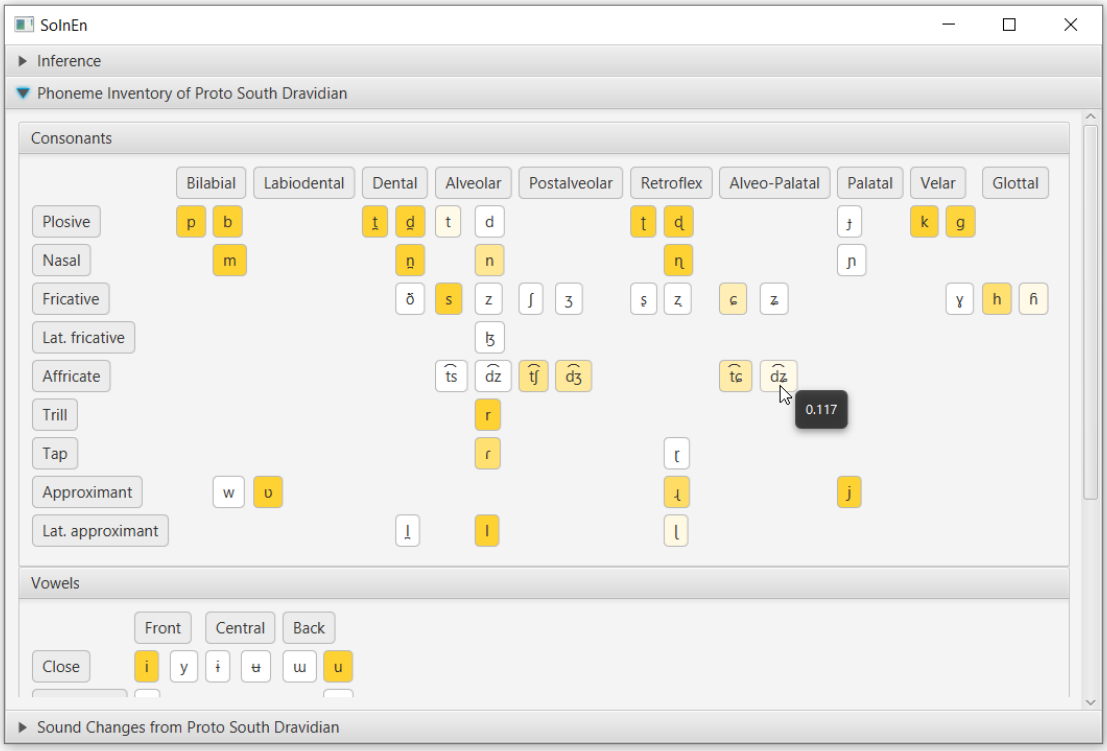

Figure 4.3: In SolnEn's second section, the inferred proto-language inventory is displayed.

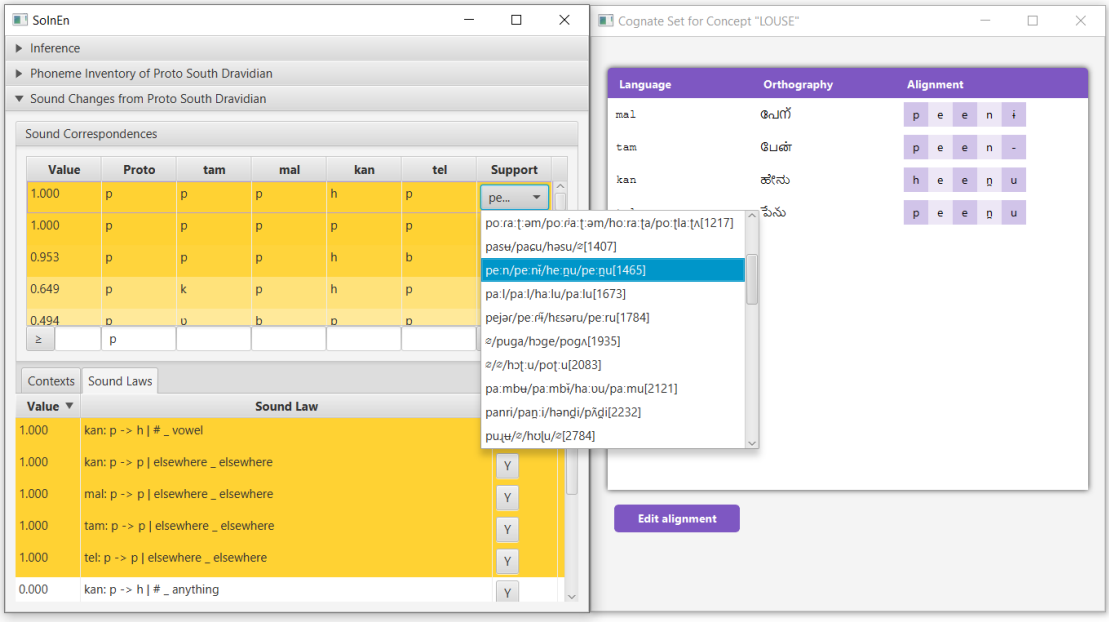

Figure 4.4: The cognate sets that provide support for a sound correspondence are displayed in the Cognate Set window upon selection.

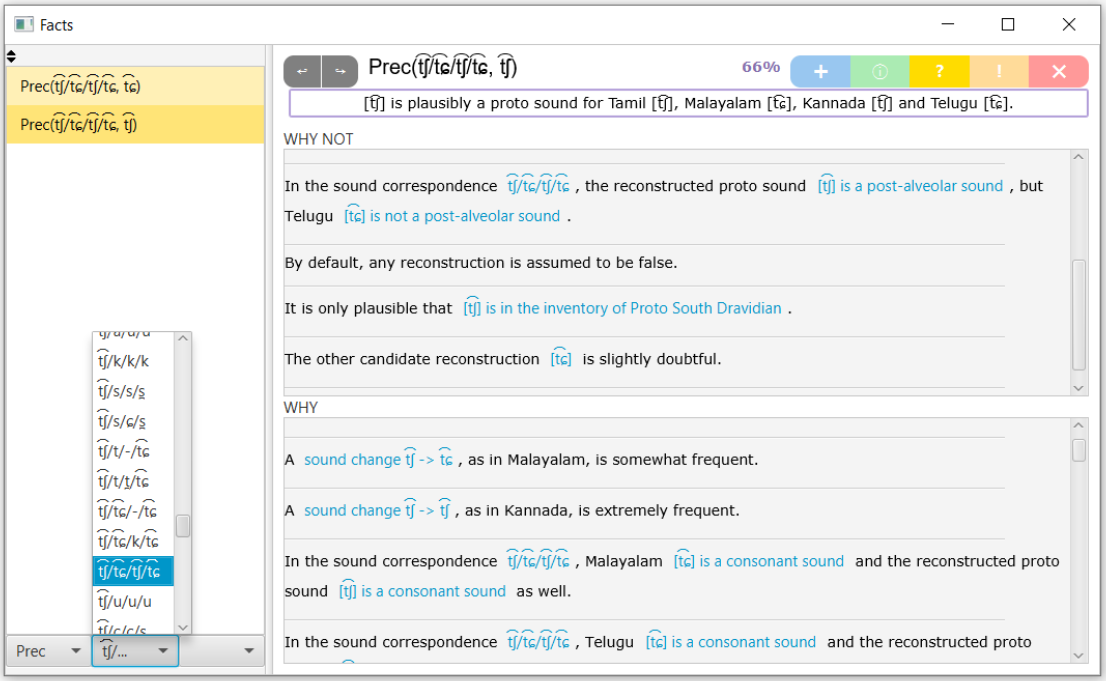

Figure 4.5: The Fact Viewer provides explanations for the inference results by verbalizing the applied ground rules.

alignments match the respective sound correspondence. Selecting a cognate set from the list changes the Cognate Set window to display the alignment of its members. This allows for easy investigation of the evidence for a sound correspondence.

The sound correspondences can be filtered via the text fields below, selecting e.g. only those correspondences for which a certain proto-sound was reconstructed. The proto-sound filter of the sound correspondences also filters the sound law display. Clicking on a proto-phoneme in the inventory pane sets the proto-sound filter as well (and opens the Sound Changes section).

Users who would like to learn how SoInEn arrived at the belief for some fact can double-click on any proto-phoneme, sound correspondence, context or sound law to open the Fact Viewer on the corresponding PSL atom. As mentioned before, the Fact Viewer provides human-readable explanations for an atom's belief value by verbalizing the ground rules containing this atom.

Figure 4.5 shows the Fact Viewer's information on the atom `Prec(tʃ͡ʃ/tɕ͡ɕ/tʃ͡ʃ/tɕ͡ɕ, tʃ͡ʃ)` (see section 4.3): Below the atom string, it first provides an explanation of the meaning of this atom. It then verbalizes ground rules that put downward pressure on the atom's belief value in the WHY NOT pane, and ground rules that apply upward pressure in the WHY pane. The rule verbalizations contain hyperlinks leading to the other atoms participating in the respective ground rule. In addition, there is a filterable list of all atoms used during inference on the left, which can be selected for inspection. The 'backward' and 'forward'

buttons to the left of the atom string aid in navigating through the rules and atoms. Finally, there are five buttons on the right of the atom string that allow the user to set an atom's belief to 1.0 or 0.0, indicate a user preference for it by means of a special prior atom, release a fixated atom for the next inference, or delete and blacklist it.

4.2 Providing World Knowledge

Some of the initial steps of the comparative method can better be done outside of PSL; partially because they are mechanical in nature and do not benefit from logical reasoning, and partially because the resulting PSL problems would be too complex in structure to be inferred efficiently, while there exist well-established algorithms to carry out the task with satisfying results. The affected steps of the comparative method are steps 1 and 2, namely cognate jugdments (section 4.2.1), alignment of the resulting cognate sets (section 4.2.2) and extracting sound correspondences from these alignments (sections 4.2.3 and 4.2.4). In addition, the model needs to be provided with linguistic knowledge about sound classes (section 4.2.5) and the general plausibility of certain sound changes (section 4.2.6).

4.2.1 Cognate Judgments

The task of detecting cognate sets actually lends itself well to PSL, since it naturally involves reasoning over several possibilities in the face of a varied collection of evidence. However, perhaps the most important kind of evidence for cognacy judgments are the genetic relationships among the involved languages and sound changes that happened between them; evidence that is the very outcome of SoInEn. Therefore, some initial cognate judgments have to be provided, and could later be refined and revised using the conclusions SoInEn has drawn from them. In fact, a model for cognacy and borrowing detection is already being designed within the EtInEn system, but is not yet at a stage where it could serve as input to SoInEn.

The NorthEuraLex data I use as input to SoInEn was enhanced with cognate set annotations that were computed using the UPGMA algorithm with similarity scores generated with the Information-Weighted Distance with Sound Correspondences (IWDSC) method, as described in Dellert (2018). IWDSC is a novel similarity score that is computed during Information-Weighted Sequence Alignment (IWSA), a modification of the well-known Needleman-Wunsch algorithm for sequence alignment that uses PMI-based phoneme similarity scores for improved word alignments.

Being an automated and not a manual expert annotation, the NorthEuraLex cognate judgments are expected to be faulty in places. One major shortcoming is that only words referring to the same concept are even considered for cognacy. Hence, related words that have shifted in meaning or belong to different parts of speech are never assumed to be cognates. Hence, SoInEn must be robust enough to draw the right conclusions in spite of noisy cognacy data.

olo	*muldu*	mʊldʊ	m	–	ʊ	–	l	d	–	ʊ
smj	*mållde*	mɔlːdɛ	m	ɔ	l	–	l	d	–	ɛ
sma	*mueltie*	mʉɛltiɛ	m	–	ʉ	ɛ	l	t	i	ɛ
hun	*föld*	føld	f	–	ø	–	l	d	–	–

Figure 4.6: Faulty alignment for the Olonets Karelian (olo), Lule Saami (smj), Southern Saami (sma) and Hungarian (hun) words for 'earth (soil)' (Dellert and Jäger 2017).

4.2.2 Alignment

Experiments with implementing multiple sequence alignment in PSL showed that the resulting dependency structure is too strongly connected for fast and scalable inference. Not only would it require up to n^k atoms to model the alignment alone, where n is the maximum length of the k involved words, but these atoms would also be heavily connected via ground rules excluding contradictive alignments and providing evidence from other sounds in the same column. Inferring alignments in PSL is therefore not feasible, which is why SoInEn relies on well-established dynamic programming algorithms to perform it externally.

For pairwise sequence alignment, I use the IWSA implementation of Dellert (2018). To combine these aligned pairs into an arbitrarily large multiple sequence alignment (MSA), I implemented the T-Coffee algorithm developed by Notredame, Higgins, and Heringa (2000)[3], which relies on the preprocessed pairwise alignment information to guide the multiple sequence alignment process for more reliable results.

Again, the automated nature of the process is sometimes prone to errors. As with the original Needleman-Wunsch alignment algorithm, insertions and deletions are especially difficult to place correctly for IWSA. Consider the faulty alignment in Figure 4.6 of some Uralic words for 'earth (soil)'. While most of the sounds are aligned properly, the additional /l/ in Lule Saami *mållde* is not correctly identified as an insertion, but grouped with the vowels of the other words.

The source of this error does neither lie in the MSA assembled by T-Coffee nor is it a design flaw in IWSA. It is caused by the phoneme similarity scores IWSA uses as substitution costs. While Needleman-Wunsch operates with gap penalties to compute costs for insertion and deletion separately from those for substitution, the inferred correspondence model IWSA draws its similarity scores from also comes with scores for individual phoneme insertion or deletion. In this model, insertion of vowels is generally cheaper than insertion of consonants (which coincides with linguistic intuition). In the pairwise alignment of Olonets Karelian *muldu* and Lule Saami *mållde*, the score for inserting /ɔ/ is -1.43 while the score for inserting /l/ is -3.36. Also, the change from /ʊ/ to /l/ gets a much lower score than /l/ insertion (-1.8), reflecting the proximity of (velarized or 'dark') /l/ to rounded high vowels. Overall, the difference in insertion costs is large enough to yield an incorrect pairwise alignment which is then propagated into the MSA in Figure 4.6.

[3]Many thanks to Gerhard Jäger who provided me with his Python implementation of the algorithm as a template.

This problem is hard to resolve, also because the error-causing similarity scores are actually justified, as explained above. It is also not within the scope of this thesis to improve IWSA. Still, the faulty alignments cause noise in the sound correspondences which are the most basic observations that will be fed to SoInEn, so the system needs to be robust enough to deal with that noise.

4.2.3 Counting Sound Correspondences

Sound correspondences were initially supposed to be inferred within the PSL model. The task seems straightforward: Sounds that are frequently aligned together are likely to constitute a regular sound correspondence. However, frequency effects are difficult to model in PSL and the possibility of languages missing in alignments further complicates the issue.

It is impossible to directly formulate a PSL rule that states "if x is frequently observed, then y". We can have a rule like this:

```
Aligned(Sounds) -> SoundCorrespondence(Sounds)
```

However, this alone will set the belief of `SoundCorrespondence(Sounds)` to 1.0 as soon as a single `Aligned(Sounds)` is observed. To counterbalance this, we would need negative evidence – but what can be evidence *against* a sound correspondence? The solution that comes closest to modeling a frequency effect would be a high negative prior on `SoundCorrespondence` that can only be overcome with enough positive evidence. The optimal weight of this will be rather data-dependent though, having to be higher when more alignments are available or when the data is expected to be noisier, and lower when there is fewer but cleaner data.

The problem of gapped alignments also still needs to be considered. While a sound correspondence consists of one sound for each target language, not all target languages might participate in every cognate set, and can therefore be missing in an alignment. The more languages we infer over, the more likely it is that alignments will be incomplete, so discarding all gapped alignments will greatly reduce the amount of evidence available. We could reformulate the above rule as follows:

```
Aligned(PartialSounds) & Compatible(PartialSounds, CompleteSounds) ->
    SoundCorrespondence(CompleteSounds)
```

What makes a gapped alignment compatible with a sound correspondence? The easiest definition is to interpret the gaps as wildcards and see if the sound correspondence can be matched by this. For example, h/p/?/p would be compatible with h/p/p/p, and ?/p/p/? with p/p/p/p. However, according to this definition, ?/p/p/? would additionally support e.g. h/p/p/p and b/p/p/b. This is problematic: No linguist would consider ?/p/p/? as evidence for these two sound correspondences, since the sound changes they display are not captured by ?/p/p/?. I therefore extend the definition of compatibility by the requirement that if the sound correspondence contains two or more different sounds, the gapped alignment must also contain two or more different sounds to be compatible.

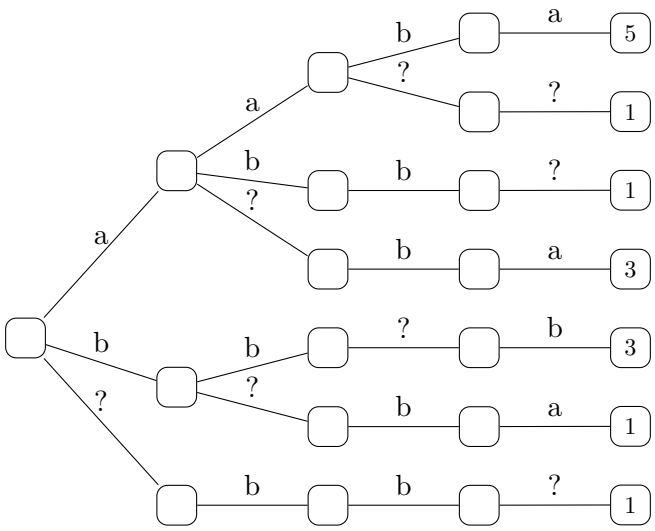

Figure 4.7: An example of a Gapped Frequency Trie.

The next issue is that not all sound correspondences might be captured by a complete alignment. We might observe b/p/p/?, b/?/p/b and ?/p/p/b, but never b/p/p/b as a whole. Still, we would like to consider b/p/p/b a possible sound correspondence. It is straightforward to introduce a *merge* operation on partial alignments with matching literals, like b/p/p/? and b/?/p/b, that fills the gaps of one alignment with the literals of the other. But what about two alignments like h/p/?/? and h/?/p/p? Is an overlap of a single literal enough to assume a sound correspondence h/p/p/p? We might want to have a parameter m, the minimum number of overlaps required for merging two partial alignments.

Finally, a partial alignment like h/p/?/? should not contribute as much evidence as a full alignment h/p/p/p to the sound correspondence h/p/p/p. The uncertainty caused by the gaps in h/p/?/? must be captured somehow. This means that either we need variable belief values of `Compatible`, assigning lower belief to alignments with more gaps, or variable rule weights depending on the number of gaps involved.

Overall, it seems exaggerated to perform this rather mathematical task in PSL, since it does not naturally involve any logical reasoning. I have therefore decided to calculate sound correspondence frequency priors in the range $[0, 1]$ outside of PSL and inject them as observations into the system.

4.2.3.1 The Gapped Frequency Trie

All alignment columns are stored and counted in a Gapped Frequency Trie, a tree-like data structure where each edge represents one sound in the alignment and the leaves contain the frequency of the (possible gapped) alignment spelled out by its path. Figure 4.7 shows an example of such a trie, where the alignments a/a/?/?, a/b/b/?, b/?/b/a and ?/b/b/? were observed once, a/?/b/a and b/b/?/b were observed three times and the complete alignment a/a/b/a was observed five times.

```
collectUngappedFrequencies(N, S, m):
    F ← x ↦ 0
    if N is leaf:
        F(S) ← N.freq
    else:
        for e in non-gap edges:
            S' ← S · e\text{.symbol}
            if e has gap edge e_g:
                S_g ← S · ?
                merge(e_g.child, e.child, S_g, S', S', F, |S|, m)
            F' ← collectUngappedFrequencies(e.child, S', m)
            F ← x ↦ F(x) + F'(x)
    return F

merge(N_1, N_2, S_1, S_2, S_c, F, l, m):
    if N_1, N_2 are leaves:
        if l ≥ m:
            if isMergeable(S_c, S_1):
                F(S_c) ← F(S_c)+ normalizedFrequency(N_1.freq, S_1)
            if isMergeable(S_c, S_2):
                F(S_c) ← F(S_c)+ normalizedFrequency(N_2.freq, S_2)
    else:
        for e_1 in N_1.edges:
            S'_1 ← S_1 · e_1.symbol
            if e_1 is a gap edge:
                for e_2 in N_2.edges:
                    if e_2 is not a gap edge:
                        S'_c ← S_c · e_2.symbol
                        S'_2 ← S_2 · e_2.symbol
                        merge(e_1.child, e_2.child, S'_1, S'_2, S'_c, F, l, m)
            else:
                S'_c ← S_c · e_1.symbol
                if N_2 has matching edge e_2:
                    S'_2 ← S_2 · e_2.symbol
                    merge(e_1.child, e_2.child, S'_1, S'_2, S'_c, F, l+1, m)
                if N_2 has gap edge e_g:
                    S'_2 ← S_2 · ?
                    merge(e_1.child, e_g.child, S'_1, S'_2, S'_c, F, l, m)

isMergeable(S_c, S_g):
    return |{c ∈ S_c}| = 1 or |{c ∈ S_g} \ {?}| ≥ 2

normalizedFrequency(f, S):
    return f · (|S| − #"?"(S))/|S|
```

Figure 4.8: The algorithm for ungapping a Gapped Frequency Trie, started by calling getUngapped-Frequencies on the root node with an empty sequence S.

To obtain the ungapped alignments and their frequencies from this, the algorithm in Figure 4.8 is used. Starting from the trie's root node, the program traverses through the trie, merging gapped alignments to compatible alignments and collecting the completed sequences together with their frequencies. It consists of two separate procedures: The traversal through the trie in `getUngappedFrequencies` and the `merge`ing of gapped subtries.

The method `getUngappedFrequencies` frames the program. It runs on a node N, the path to which is the sequence (or alignment) S. The parameter m refers to the minimum number of overlaps discussed before. In SoInEn, it can be set by the user, but has a default value of $m = \max(2, \lceil \frac{n}{2} \rceil)$ where n is the number of target languages. Our return value is the function F which maps a complete alignment to its frequency.

`GetUngappedFrequencies` is only called along non-gap edges, so when N is a leaf, we know that S is a complete alignment, and we can inject its frequency stored in N into F. Otherwise, we loop through its non-gap edges. For each edge e, we extend a copy S' of the path S by the symbol labeling the edge. If N has a gap edge e_g, we attempt to `merge` the subtrie rooted by the child of gap edge e_g to the subtrie rooted by the child of e. Then we recurse on e's child and the extended sequence S'.

The `merge` method traverses two subtries in parallel, rooted by nodes N_1 and N_2, with S_1 and S_2 being the sequences or paths leading to the respective nodes. S_c is the completed sequence of the two where the gaps of one have been filled in by the literals of the other. Finally, l refers to the number of literal matches so far. `Merge` is originally called by `getUngappedFrequencies` on the children of e_g and e and their respective paths. S_c is initialized as the path to e's child, because it is guaranteed to contain no gaps. l then is the length of S, since until N, only literals were encountered on the path.

If `merge` has reached the bottom of the subtries, it is time to check the compatibility criteria: Has the minimum number of literals m been matched ($l \geq m$)? Are any of the two potentially gapped sequences S_i ($i \in \{1, 2\}$) compatible with the completed sequence S_c, i.e. does S_c consist of all equal symbols or does S_i contain at least two different symbols? If yes, the frequency $F(S_c)$ of the complete sequence S_c is updated by adding the normalized frequency of S_i. The normalization is to account for the insecurity that comes with being gapped, and is applied by scaling S_i's raw frequency by the percentage of literals in the sequence ($\frac{|S_i| - \#_{\sim r}(S_i)}{|S_i|}$). Thus, alignments with more gaps contribute less to the overall frequency of a completed alignment.

If N_1 and N_2 are not leaves, we loop through the outgoing edges of N_1, extending S_1 by the respective edge's symbol. If it is a gap edge, we recursively `merge` its subtrie to the subtries of N_2's non-gap edges. Otherwise, it is `merge`d to N_2's gap subtrie (if existent) and N_2's subtrie with matching symbol (if existent).

Via this procedure with $m = 1$, we will finally arrive at the following completed alignments and frequencies for the example in Figure 4.7:

```
F(a/a/b/a) = 7.25   (a/a/b/a: 5 * 1.0, a/?/b/a: 3 * 0.75)
F(a/b/b/a) = 3.0    (a/b/b/?: 1 * 0.75, a/?/b/a: 3 * 0.75)
F(b/b/b/b) = 2.75   (b/b/?/b: 3 * 0.75, ?/b/b/?: 1 * 0.5)
```

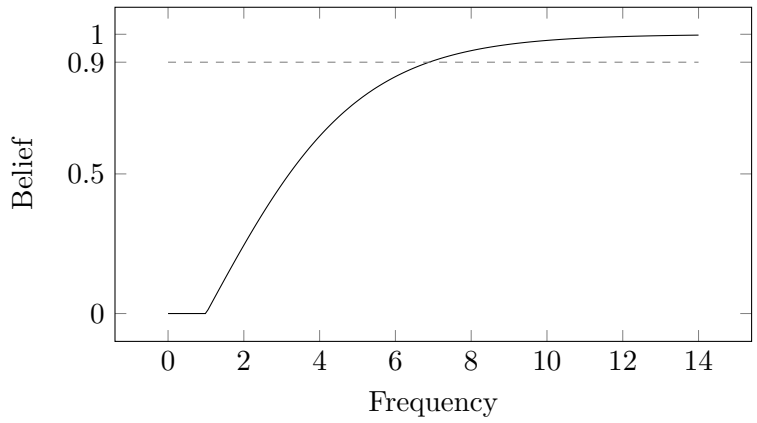

Figure 4.9: The function used to convert sound correspondence frequencies into belief values.

The gapped alignments a/a/?/? and b/?/b/a could not be merged into any completed alignment. Also, a/a/?/? and ?/b/b/? could not be counted as evidence for a/a/b/a and a/b/b/a, respectively, since they did not reflect the symbol variation in the completed alignments. When setting $m = 2$, the values for a/a/b/a and a/b/b/a are the same, but the completion b/b/b/b is not included, since the overlap between b/b/?/b and ?/b/b/? is not large enough to merge them.

To convert these frequencies to belief values in the range $[0, 1]$, I apply the upper half of a logistic function. The logistic function is a simple sigmoid (S-shaped) function with the equation

$$f(x) = \frac{M}{1 + e^{-k(x-x_0)}} \tag{4.1}$$

where M is the function's maximum, x_0 is the midpoint, and k determines the steepness of the curve. The monotonically increasing, concave shape of the logistic function's upper half has the effect that the resulting belief saturates after some frequency threshold, implementing the idea that if an alignment has been observed a certain number of times, it can be safely assumed that it is indeed a regular sound correspondence.

For the sound correspondence frequencies, I use the variant

$$f(x) = \max(0, \frac{2}{1 + e^{-0.5(x-1)}} - 1) \tag{4.2}$$

which results in the curve displayed in Figure 4.9. An alignment must be observed more than once to be even considered for sound correspondence status and only saturates (reaches a belief over 0.9) after a frequency of ~ 6.89. In the above example with $m = 1$, we get a belief of ~ 0.92 for a/a/b/a, ~ 0.46 for a/b/b/a, and ~ 0.41 for b/b/b/b.

4.2.4 N-Grams of Sound Correspondences

In addition to the frequencies of single sound correspondences, we also need to collect the frequencies of sequences or n-grams of sound correspondences as evidence for conditional

tam	*maram*	mʌɾʌm	m	ʌ	ɾ	–	ʌ	m
mal	*maraṁ*	maɾʲam	m	a	ɾ	ʲ	a	m
kan	*mara*	məra	m	ə	r	–	a	–
tel	–	–	?	?	?	?	?	?

Figure 4.10: Dravidian alignments of the NorthEuraLex cognate set containing the Tamiḻ (tam), Malayāḷaṁ (mal) and Kannaḍa (kan) words for 'tree' (Dellert and Jäger 2017). Telugu (tel) has no cognate inside NorthEuraLex.

sound laws, and these need to be transformed into belief values as well.

So while reading in alignments, we also collect bigram and trigram frequencies of alignment columns, padded with the wound boundary symbol #. From the Dravidian alignment in Figure 4.10 (repeated and extended from Figure 1.1) for instance, we would get the following alignment n-grams:

Bigrams:	*Trigrams:*
(#/#/#/?, m/m/m/?)	(#/#/#/?, m/m/m/?, ʌ/a/ə/?)
(m/m/m/?, ʌ/a/ə/?)	(m/m/m/?, ʌ/a/ə/?, ɾ/ɾ/r/?)
(ʌ/a/ə/?, ɾ/ɾ/r/?)	(ʌ/a/ə/?, ɾ/ɾ/r/?, –ʲ/–/?)
(ɾ/ɾ/r/?, –ʲ/–/?)	(ɾ/ɾ/r/?, –ʲ/–/?, ʌ/a/a/?)
(–ʲ/–/?, ʌ/a/a/?)	(–ʲ/–/?, ʌ/a/a/?, m/m/–/?)
(ʌ/a/a/?, m/m/–/?)	(ʌ/a/a/?, m/m/–/?, #/#/#/?)
(m/m/–/?, #/#/#/?)	

After the Frequency Trie containing the simple sound correspondence frequencies has been ungapped, the gapped alignments in the n-grams are replaced by the completions generated by the trie to receive sound correspondence n-grams. As with the ungapped alignment frequencies, the raw n-gram frequencies are scaled by the percentage of literals present in the original n-gram. If, for example, ʌ/a/a/? were completed as both ʌ/a/a/a or ʌ/a/a/ʌ, m/m/–/? as m/m/–/m, and #/#/#/? as #/#/#/#, the final alignment trigram in the above list would yield the two sound correspondence n-grams (ʌ/a/a/a, m/m/–/m, #/#/#/#) and (ʌ/a/a/ʌ, m/m/–/m, #/#/#/#), both with frequency $\frac{3}{4}$.

To convert the obtained frequencies into belief values, we again use the upper half of a logistic function. Since bigram and trigram data naturally is more sparse than the unigram data collected in the trie, we set the steepness k to 1 (instead of 0.5) for quicker saturation and the midpoint x_0 to 0 to already take n-grams with raw frequency 1 or less into account. The resulting function is

$$f(x) = \frac{2}{1 + e^{-x}} - 1 \tag{4.3}$$

which, as illustrated in Figure 4.11, saturates (reaches belief ≥ 0.9) already at frequency ~ 2.99.

4.2.5 Sound Classification

To estimate the likelihood of a certain sound change and to produce generalized contexts for sound change, SoInEn requires a theory of sound classes and features: It needs to

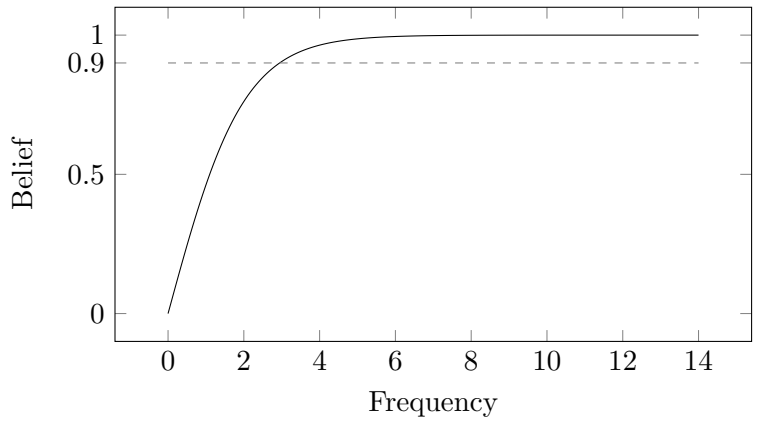

Figure 4.11: The function used to convert sound correspondence n-gram frequencies into belief values.

know what a vowel or a fricative is, and that rounded vowels contrast with unrounded vowels.

To retrieve these sound classes, I first extracted the features for all 122 IPA graphemes occurring in NorthEuraLex from the Cross-Linguistic Transcription Systems (CLTS) database (List, Anderson, et al. 2019; Anderson et al. 2018). These consist of voicedness, manner and place of articulation (plus, if applicable, lateral or sibilant annotation) for consonants, and roundedness, horizontal and vertical position for vowels. In analogy to the laterals, I added the rhotic feature to all 'r-like' consonants. All phonemes can be unambiguously referred to by a combination of these basic features.

The resulting basic features were then grouped together into superclasses that historical linguists would commonly like to refer to, such as liquids, continuants or sonorants. These superclasses are defined in terms of the subclasses they can be partitioned into, such that each sound in the superclass belongs to exactly one partitioning subclass. They can be defined multiple times if there are several possible partitionings. The class of vowels, for instance, can be partitioned according to the following criteria:

Roundedness: rounded, unrounded

Horizontal position: front, near-front, central, near-back, back

Vertical position: open, near-open, open-mid, mid, close-mid, near-close, close

These partitions allow reasoning about sound classes, for example when inferring a general context for some sound change: Competing sound classes contradict each other (e.g. the sound change cannot have happened next to unrounded vowels if it was already concluded that it happened next to rounded vowels), and the observation of several competing subclasses suggest the superclass context (e.g. the observation of rounded and unrounded vowels should entail the general context 'vowel').

This subclass-superclass structure implies a hierarchy of sound classes, which are made explicit by assigning each sound class a level following their topological order, such that every sound class has a lower level than all of its subclasses. Overall, this leads to five

levels, where level 1 consists only of the 'anything' class that contains all sounds, defined by the contrasts vowel vs. consonant or obstruent vs. sonorant, and level 5 contains all the basic features extracted from CLTS.

The full list of sound classes and partitions can be found in appendix B.

4.2.6 Sound Transition Matrix

As mentioned in section 1.3.1.3 when discussing the comparative method, reconstruction decisions are usually also guided by the linguist's knowledge about what is a common, plausible sound change. Since this directly depends on the reconstructions for other languages, we cannot infer this knowledge inside PSL, or at least we need some initial ideas to begin with. Unfortunately, there is no comprehensive gold standard list of common sound changes among the language families of the world, so I needed to generate such a list myself. The result is a sound transition matrix with transition frequency scores in the interval $[0, 1]$ for each possible sound change between the $n = 122$ IPA symbols occurring in the NorthEuraLex database.

In order to count sound change frequencies, we need sound changes, and for this, we need proto-sound reconstructions, the very goal of SoInEn's inference. To approximate a history of sound changes, I used an implementation of the Sankoff algorithm by Johannes Dellert. Sankoff successively reconstructs ancestral word forms along the edges of a language tree, where the leaves point to the aligned word forms of the modern languages, yielding a number of sound changes from each node in the tree to its child nodes. As input, I used the MSAs that are also the basis for the sound correspondences (cf. section 4.2.2) and the Glottolog 3.0 tree reduced to the languages in NorthEuraLex (Dellert 2017).

The naive approach to generate transition scores $\tau(s, t)$ for the sound change from a source sound s to a target sound t from the resulting raw sound change frequency $f(s, t)$ would be to use the percentage of sound changes $s \to t$ among all sound changes from s:

$$\tau(s,t) \stackrel{?}{=} \frac{f(s,t)}{\sum_{i=1}^{n} f(s,t_i)} \tag{4.4}$$

This poses a problem: In the vast majority of cases, the sound remains unchanged. For instance, /t/ changed to itself in 91.46%, /f/ in 94.86% and /g/ in 87.15% of all cases. Still, there are real sound changes for these sounds that are more frequent than others, and this should be reflected in the belief values. However, most transition percentages from sounds other than the targets itself range between 2% and 0.1% (some even lower), all of which effectively constitute a belief value of 0. /d/, for example, is a target sound of /t/ in only 1.11% of cases; directly translated into belief values, a sound change from /t/ to /d/ would be virtually prohibited, even though linguists would certainly want this common sound change to receive a high belief.

We would like to assign high belief to sound changes that occur frequently, but we do not necessarily need to distinguish more frequent changes from less frequent changes above a certain frequency threshold. For instance, it would be fine to assign a belief of 1 to both

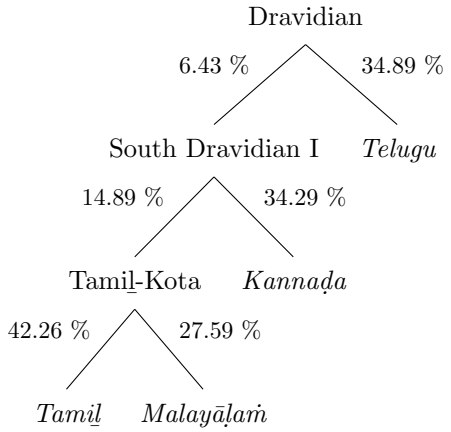

Figure 4.12: NorthEuraLex' Dravidian languages in the reduced Glottolog tree with innovativeness values.

/t/ → /t/ and /t/ → /d/, since both are plausible sound changes, even though /t/ → /t/ is notably more frequent than /t/ → /d/. To balance out the amount of identity transitions and true sound changes a little bit, we might also want to put more weight on sound changes observed in languages that are generally more innovative, in that have undergone more actual sound changes from their parent language.

4.2.6.1 Innovativeness

Let $f_L(s,t)$ be the number of sound changes from s to t into a languge L from its direct parent in the Glottolog tree. The *innovativeness* $\nu(L)$ of that language then is the percentage of true sound changes:

$$\nu(L) = \frac{\sum_{s \neq t} f_L(s,t)}{\sum f_L(s,t)} \tag{4.5}$$

The resulting value is a straightforward indicator of how much the daughter language differs from its parent phonologically. However, it is also rather dependent on the languages' phonetic transcription, the language coverage of the underlying database (in our case NorthEuraLex), the alignments and the resulting reconstructions of the Sankoff algorithm.

Consider, for instance, the innovativeness measures of the Dravidian languages displayed in Figure 4.12. Here, Tamiḻ seems to be the most innovative language, even though it is actually phonologically quite conservative. Malayāḷaṁ, on the other hand, seems to be the least innovative, but as discussed in section 3.1.5, it has most generously adapted Sanskrit phonemes.

One of the problems with Tamiḻ here is that the phonetic transcription in NorthEuraLex, rather than being phonemic, sticks close to actual pronunciation. Therefore, short /a/, for instance, is (in version 0.9) transcribed as [a] in Malayāḷaṁ, as [ʌ] in Tamiḻ and Telugu,

and as [ə] or [a] (depending on the position in the word) in Kannaḍa. This leads to words containing /a/ often not being reconstructed as [ʌ] by Sankoff, producing a sound change into Tamil̲ where there actually is none.

A general issue with conservative languages is that they usually stick out phonetically, having resisted sound change while their siblings have drifted away from them. Since Sankoff knows nothing about the conservativity of these languages, it will often go with the majority vote and reconstruct according to the phonemes of the less conservative siblings, leading to many sound changes into the conservative language. Icelandic, for example, also has an extremely high innovativeness score of 50.46%, despite generally being regarded as quite conservative.

Finally, the Sankoff algorithm can only reconstruct on the basis of the languages it is given. While NorthEuraLex has a good coverage for the languages of Northern Eurasia, especially Uralic, it contains only 4 Dravidian languages, all of them belonging to the same branch (South Dravidian). With more Dravidian languages weighing in on the reconstruction, Sankoff would probably perform better and give a more accurate innovativeness score for Tamil̲.

In general, however, the innovativeness measures approximately reflect linguistic intuition, assigning high scores to known "quirky" languages like English (47.28%), French (40.5%) or Hungarian (47.94%), and lower scores to languages known to be more conservative within their subfamily, such as Finnish (22.3%), North Karelian (12.46%) or Slovenian (20.23%).

4.2.6.2 Converting Normalized Frequencies to Belief

As usual, I employ the upper half of a logistic function to derive belief values from the raw frequency counts. To calculate the transition score $\tau_L(s,t)$ for a language L, I apply the logistic function to the raw frequency scaled by the language's innovativeness:

$$\tau_L(s,t) = \frac{2}{1+e^{-\nu(L)f_L(s,t)}} - 1 \tag{4.6}$$

Calculating language-specific scores first has the advantage of providing a "normalized frequency" for each language. The overall number of sound changes $f_L(s,t)$ observed can vary greatly between languages, since it depends on the number of entries for a language as well as the number of found cognates sets and, ultimately, the average word length (languages with longer words yield more sound changes). $\tau_L(s,t)$, on the other hand, lies within the interval $[0,1]$ for every language.

To arrive at the general transition score $\tau(s,t)$, I then apply the logistic function again to the sum of the language-specific transition scores:

$$\tau(s,t) = \frac{2}{1+e^{-0.1\sum\tau_L(s,t)}} - 1 \tag{4.7}$$

The resulting transition scores reflect our intuition about plausible sound changes quite

well. The score for /t/ → /d/, for instance, is 0.97, supporting this sound change almost as much as the identity transition /t/ → /t/ (score 1.0). See Appendix A for a link to the complete matrix.

4.3 Phase 1: Proto Inventory Reconstruction

Having covered the necessary pre-processing steps performed outside of PSL, I now turn to the design of the actual PSL model. The first inference phase implements step 3 of the comparative method: The reconstruction of proto-sounds for the sound correspondences (cf. section 1.3.1.3). The input observations for this phase are the sound correspondences collected from the alignments as well as the knowledge about sound classes and plausible sound changes extracted from the sound class hierarchy and the sound transition matrix. There are two targets to this inference phase: The proto-sound reconstructions for the individual sound correspondences and the overall phoneme inventory of the proto-language.

4.3.1 Predicates

> **Pcor**(SoundCorrespondenceID)
> "The sound correspondence SoundCorrespondenceID is frequent."
> **Pcos**(SoundCorrespondenceID, Language, LanguageSound)
> "In the sound correspondence SoundCorrespondenceID, the reflex in Language is LanguageSound."

These two predicates model the sound correspondences extracted during pre-processing. The `Pcor` predicate's belief value is the frequency prior calculated as described in section 4.2.3. The actual sounds a correspondence is composed of can be accessed via the `Pcos` predicate, whose only possible belief value is therefore 1.0. The `SoundCorrespondenceID` can in principle be any kind of ID; in SoInEn, however, the actual sound correspondence string `Sound1/.../SoundN` (e.g. `h/p/p/p`) is used as id for user readability.

> **Psub**(Sound, SoundClass)
> "Sound is in SoundClass."

`Psub` is the first of three predicates representing the sound class hierarchy (but the only one needed in this inference phase), implementing the subset-or-equals relation. Additionally, when the first element is a phoneme, atoms are grounded for all possible combinations of phoneme and sound class, linking sounds to each of their superclasses with belief 1.0 and to all other classes with belief 0.0. This is necessary for rule `pcorToPrecDiffClass`. For proper sound classes, the 0.0-belief relations are not grounded. For instance, the following ground atoms will be set to 1.0 for /t/:

```
Psub("t", "t")
Psub("t", "voiceless")
Psub("t", "alveolar")
Psub("t", "stop")
Psub("t", "occlusive")
Psub("t", "obstruent")
Psub("t", "consonant")
Psub("t", "anything")
```

```
generateProtoCandidates(S, C = {C_1, ..., C_n}, k):
    P ← S
    C' ← {C_i ∈ C | S ⊆ C_i}
    if |C'| ≥ k:
        P ← P ∪ getSimilarSounds({C_i ∈ C \ C' | C_i ∩ S ≠ ∅}, ⋂_{C_i ∈ C'} C_i)
    return P

getSimilarSounds(C_r, B):
    P_s ← ∅
    for C'_r in P(C_r):
        B_e ← B ∩ ⋂_{C_i ∈ C'_r} C_i
        if |B_e| = 1:
            P_s ← P_s ∪ B_e
    return P_s
```

Figure 4.13: The algorithm for generating proto-sound candidates for a sound correspondence consisting of sounds S, given the sound classes C.

All other combinations of `"t"` and a sound class are set to 0.0.

> **`Pchg`**`(SourceSound, TargetSound)`
> "The sound change from `SourceSound` to `TargetSound` is common."

The belief value of `Pchg` is the value given in the sound transition matrix described in section 4.2.6. The higher this value, the more common and plausible is a sound change between the two arguments.

> **`Prec`**`(SoundCorrespondenceID, ProtoSound)`
> "The sound correspondence `SoundCorrespondenceID` evolved from `ProtoSound`."

`Prec` is one of the two target predicates in this phase and models the proto-phoneme reconstruction for a sound correspondence.

> **`Pinv`**`(Language, Sound)`
> "`Sound` is in `Language`'s phoneme inventory."

In general, `Pinv` models the phoneme inventories of all participating languages. For the observed modern languages, `Pinv`'s sound argument and belief are fixed, but with the proto-language as first argument, it is the second target predicate of this inference phase, representing the PSL model's belief that a certain sound occurred in the proto-inventory.

4.3.2 Ideas

Plausible `Prec` atoms, i.e. proto-sound candidates for a sound correspondence, are generated using the algorithm shown in Figure 4.13. As input, the method receives the unique phonemes S of the sound correspondence, the sound classes C (where each sound class $C_i \in C$ is a set of sounds) and the minimum number of shared sound classes k. By default, k is set to 1, but it can be raised by the user to only generate unattested proto-sounds that

share many features with all of the attested sound, e.g. in case noisy data could provide high support for unsuitable generated proto-sounds.

A set of proto-candidates P minimally consists of the sound correspondence's members S themselves. To see whether additional feasible candidates can be found, we consider the set C' of lowest-level sound classes shared by all sounds in S: If C' contains less than k classes, the sounds of the sound correspondence are too dissimilar to generate a reasonable number of additional candidates. With $k = 1$, for instance, the correspondence m/t/t has too few shared sound classes (namely none), while m/d/d has just enough (`voiced`) to reach the next step.

If C' passes the threshold, the algorithm tries to compose additional candidates from the base sounds $B = \bigcap_{C_i \in C'} C_i$ in the fully shared classes and the remaining partially shared classes C_r. For this, it filters the base sounds B for each possible combination of partially shared classes C'_r in C_r (as spelled out by the power set $\mathcal{P}(C_r)$), yielding the sounds B_e which are in the classes C' as well as C'_r. If B_e contains only a single sound, meaning that this sound can be completely specified by a combination of sound classes occurring in the sound correspondence, this sound is added to the set of candidates.

For the sound correspondence m/d/d, for instance, we have $C' = \{\texttt{voiced}\}$, therefore, B contains all voiced consonants. C_r then consist of all other sound classes of /m/ and /d/, so $C_r = \{\texttt{bilabial}, \texttt{nasal}, \texttt{alveolar}, \texttt{stop}\}$. Possible subsets of C_r that can extend C' to a full sound specification are $\{\texttt{bilabial}, \texttt{nasal}\}$, $\{\texttt{bilabial}, \texttt{stop}\}$, $\{\texttt{alveolar}, \texttt{nasal}\}$ and $\{\texttt{alveolar}, \texttt{stop}\}$, yielding the proto-sound candidates /m/, /b/, /n/ and /d/.

`Pinv` atoms for the proto-language are inserted for all proto-candidates generated for some sound correspondence.

4.3.3 Rules

> ***precPrior:***
> `5.0: ~Prec(ID, Proto)`
> "By default, any reconstruction is assumed to be false."

This negative prior on proto-sound reconstructions for sound correspondences ensures that only those reconstructions with sufficient positive evidence are assigned high belief values. The weight for this prior is very high because of the next two rules, which can provide a rather large amount of evidence. If the prior weight was lower, they could overpower it too easily.

> ***pcorToPrecSameClass:***
> `0.25: Pcos(ID, Lang, LangSound) & Psub(Proto, Class) &`
> `  Psub(LangSound, Class) & XPrec(ID, Proto) -> Prec(ID, Proto)`
> "If some target sound and the proto-candidate belong to the *same* sound class, it is *more* likely that the candidate is indeed the proto-sound for this sound correspondence."
> ***pcorToPrecDiffClass:***
> `0.25: Pcos(ID, Lang, LangSound) & Psub(Proto, Class) &`

```
~Psub(LangSound, Class) & XPrec(ID, Proto) -> ~Prec(ID, Proto)
```
"If some target sound and the proto-candidate belong to *different* sound classes, it is *less* likely that the candidate is indeed the proto-sound for this sound correspondence."

These two rules implement the idea that a sound does not suddenly become an entirely unrelated phoneme, but that the target sounds bear some resemblance to their common ancestor, and that a proto-reconstruction becomes more likely when it shares many features with its descendants. They also indirectly implement the "majority vote" (cf. section 1.3.1.3): When proto- and target sound are equal, they belong to the exact same sound classes. The second rule actually shows a direct implementation of negative evidence, which is possible because the `Psub` atoms could be fully spelled out due to the number of phonemes and sound classes being limited to a manageable amount.

pchgPrior:
```
0.5: Pcos(ID, Lang, LangSound) & Pchg(Proto, LangSound) & XPrec(ID,
    Proto) -> Prec(ID, Proto)
```
"If the change from a proto-candidate to one of the target sounds is a common sound change, it is more likely that the candidate is indeed the proto-sound for this sound correspondence."

Here, the values from the previously generated sound transition matrix (cf. section 4.2.6) are used to support common sound changes.

singleProtoSound:
```
Prec(ID, +Proto) <= Pcor(ID) .
```
"A sound correspondence can be derived from at most one proto-sound."

This arithmetic rule enforces two restrictions at the same time. Primarily it ensures that there is at most one proto-sound selected (or multiple with low belief) for any sound correspondence, since `Pcor(ID) <= 1`. Additionally, it takes the overall frequency of the sound correspondence as encoded in the `Pcor` atom into account, making reconstructions for any low frequency sound correspondence less plausible. This sorts out dubious sound correspondences most likely the result of alignment errors and prevents their proto-reconstructions from cluttering the inferred proto-sound inventory.

protoSoundPrior:
```
1.0: ~Pinv("ProtoLang", ProtoSound)
```
"By default, all sounds are assumed to not have been part of the proto-language's sound inventory."

This rule puts a negative prior on those `Pinv` atoms that refer to the currently inferred proto-language, so that the system does not assume the existence of a proto-sound for which it has no or little evidence.

protoSoundInventory:
```
2.0: XPinv("ProtoLang", ProtoSound) & Prec(ID, ProtoSound) ->
    Pinv("ProtoLang", ProtoSound)
```

> "If a sound has been reconstructed as the ancestor of a sound correspondence, it must have been in the proto-language's sound inventory."

Finally, the `Pinv` atoms of the proto-language are supported by all ancestral sound reconstructions for the sound correspondences. The weight is higher than that of the negative prior since even a reconstruction for a single sound correspondence entails that this sound must have been part of the proto-sound inventory.

4.4 Phase 2: Context Detection

The second inference phase partially implements step 4 of the comparative method, namely the detection of possible environments for conditioned sound change (cf. section 1.3.1.4). Using the sound correspondence n-grams observed in the alignments (cf. section 4.2.4) and the proto-sound reconstructions for these sound correspondence inferred in the previous phase, this phase bundles the observed neighbors of each sound correspondence up into a single sound class context (possibly `anything _ anything`) in which it occurs, forming the basis for the conditional sound laws inferred in the final phase.

Before this phase starts, the `Pinv` atoms with belief < 0.6 as well as all `Prec` atoms reconstructing one of these proto-sounds are deleted, and the remaining `Pinv` and `Prec` atoms are fixated.

4.4.1 Predicates

> **`Pdsb`**`(SoundClass, SuperClass)`
> "`SoundClass` is the direct subclass of `SuperClass`."
> **`Plvl`**`(SoundClass, Level)`
> "`SoundClass` is at rank `Level` in the sound class hierarchy."

These two predicates define the sound class hierarchy: `Pdsb` links child classes to their direct parent classes, while `Plvl` links each sound class to its numeric level in the hierarchy, as described in section 4.2.5.

> **`Pccl`**`(SoundCorrespondenceID, SoundClass)`
> "A likely proto-sound candidate of the sound correspondence `SoundCorrespondenceID` is in `SoundClass`."

The proto-sound of a sound correspondence `SoundCorrespondenceID` is assumed to belong to a sound class `SoundClass` if there exists a `Prec(SoundCorrespondenceID, ProtoSound)` with belief > 0.5 where `ProtoSound` is in `SoundClass`. The belief of any `Pccl` is always 1.0. It is just a helper atom used in filter clauses.

> **`Pnei`**`(LeftNeighborID, TargetID, RightNeighborID)`
> "The sound correspondence `TargetID` can frequently be observed between the sound correspondences `LeftNeighborID` and `RightNeighborID`."

`Pnei` atoms carry the sound correspondence n-grams extracted as described in section 4.2.4 with their respective frequency indicators as belief values. In bigrams, the empty

neighbor is labeled "unknown". To keep the number of Pnei atoms reasonable, only those n-grams with a frequency indicator > 0.2 are converted into atoms, and only if the middle sound correspondence contains at least two different sounds. Sound "changes" where the proto-sound is retained in all daughter languages will later be considered the default "fallback" case for instances where no conditioned sound change can be applied (or when none exists).

> **Pcxt**(LeftProtoContext, TargetID, RightProtoContext)
> "The sound correspondence TargetID developed between the proto-sound (classes) LeftProtoContext and RightProtoContext."

While the neighbors of Pnei atoms are sound correspondences, the contexts of this phase's sole target predicate Pcxt are proto-sounds or sound classes. They encode the site of a sound change that produced the sound correspondence in question.

4.4.2 Ideas

Populating Pcxt is rather straightforward: For each Pnei(LNei, T, RNei), Pcxt atoms are inserted for any possible combination of sound classes of plausible (i.e. belief > 0.5) proto-sound reconstructions for LNei and RNei. For instance, let's say we have the following atoms:

```
Pnei("a/a/a", "d/t/t", "u/i/u") = 0.85
Prec("a/a/a", "a") = 1.0
Prec("u/i/u", "u") = 0.7
Prec("u/i/u", "i") = 0.3
Prec("u/i/u", "y") = 0.0
```

The only plausible proto-reconstruction for the left neighbor a/a/a is a, and for the right neighbor u/i/u, it is u. Hence, the following Pcxt atoms will be injected:

```
Pcxt("a", "d/t/t", "u")
Pcxt("a", "d/t/t", "rounded")
Pcxt("a", "d/t/t", "close")
Pcxt("a", "d/t/t", "back")
Pcxt("a", "d/t/t", "vowel")
Pcxt("a", "d/t/t", "sonorant")
Pcxt("a", "d/t/t", "anything")
Pcxt("open", "d/t/t", "u")
```

```
Pcxt("open", "d/t/t", "rounded")
Pcxt("open", "d/t/t", "close")
Pcxt("open", "d/t/t", "back")
Pcxt("open", "d/t/t", "vowel")
Pcxt("open", "d/t/t", "sonorant")
Pcxt("open", "d/t/t", "anything")
etc.
```

An "unknown" argument in Pnei remains "unknown" in Pcxt. In addition, every sound correspondence PcorID receives a Pcxt("unknown", PcorID, "unknown") in case no plausible context can be found.

4.4.3 Rules

Speaking more generally, this phase implements the fusion of single sound observations to sound classes that are just general enough to match all observations. This is where

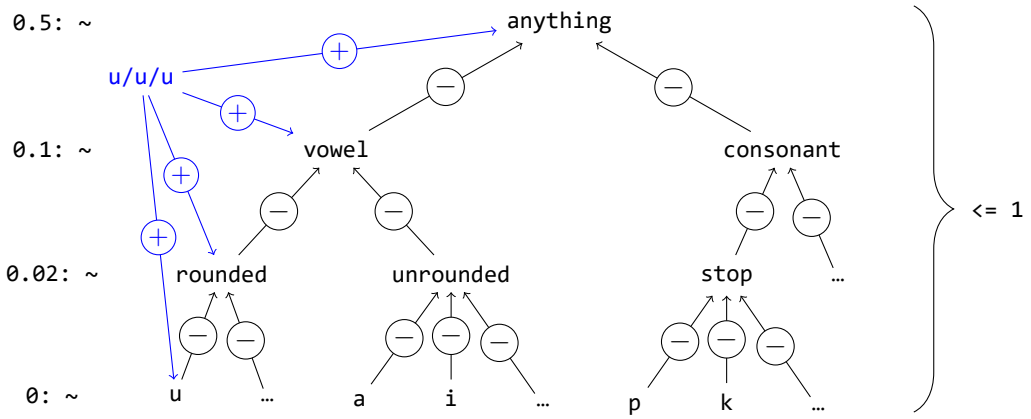

Figure 4.14: Effects of the context detection rules.

the tiered sound classes from section 4.2.5 come into play. Figure 4.14 shows a simplified excerpt of the sound class tree that illustrates how the following PSL rules push the contexts into place without any overt negative evidence.

```
levelPriorPcxt:
0.5 / 5^(lvl - 1): Plvl(L, "lvl") & XPcxt(L, Trgt, R)
   -> ~Pcxt(L, Trgt, R)
0.5 / 5^(lvl - 1): Plvl(R, "lvl") & XPcxt(L, Trgt, R)
   -> ~Pcxt(L, Trgt, R)
```

"The more general a context, the less likely it is by default."

Since there is no explicit negative evidence for n-grams, negative priors are required instead. These get weaker the lower down in the sound class tree we are: As we can see in Figure 4.14, the strongest negative prior is on the `anything` class, the second strongest on `vowel` and `consonant`[4], and so on, while the leaves, i.e. the "classes" containing only a single sound, have no negative prior.

This ensures that classes are not propagated up without good reason: When there is equal evidence for a class and its subclass, the subclass will get a higher belief. For example, if we have observed only rounded vowels, we would like the system to infer the `rounded` class and not `vowel`, even though all of the observed sounds are also vowels. The priors are sufficiently low for the first unrounded vowel to provide enough extra `vowel` evidence for the system to discard the `rounded` hypothesis.

```
singleContext:
1.0: Pcxt(+L, Trgt, +R) <= 1
```

"A sound correspondence can occur in at most one context."

This is a lightly weighted rule and not a constraint because it is technically possible for a sound correspondence to have multiple valid contexts: For example when a sound change

[4]In fact, there is a stage between `anything` and `vowel`/`consonant`, namely the `sonorant`/`obstruent` contrast, which was left out in the example tree for the sake of simplicity.

happened both at the end of words and before consonants. Still, we would like to give the system some incentive to settle on one context in most cases.

> ***contextObservation:***
> `1.0: Pcxt(+LCls, Trgt, "unknown") >= Pnei(LNei, Trgt, "unknown")`
> `{LCls: Pccl(LNei, LCls)}`
> `1.0: Pcxt("unknown", Trgt, +RCls) >= Pnei("unknown", Trgt, RNei)`
> `{RCls: Pccl(RNei, RCls)}`
> `1.0: Pcxt(+LCls, Trgt, +RCls) >= Pnei(LNei, Trgt, RNei)`
> `{LCls: Pccl(LNei, LCls)} {RCls: Pccl(RNei, RCls)}`
> "The observation of a sound correspondence n-gram supports one of the matching contexts."

These rules distribute the belief mass of each of the bigram and trigram observations among all matching contexts. This is illustrated in blue in Figure 4.14: Here, the observation of the sound correspondence `u/u/u` next to the target supports the contexts `u`, `rounded`, `vowel` and `anything`. Which of these contexts is chosen in the end depends on which sound classes are supported by the other observations and how these are propagated up by the other (restrictive) rules.

> ***notSupergroupPcxt:***
> `1.0: Pcxt(LCls, Trgt, R) & Pdsb(LCls, LGrp) -> ~Pcxt(LGrp, Trgt, R)`
> `1.0: Pcxt(L, Trgt, RCls) & Pdsb(RCls, RGrp) -> ~Pcxt(L, Trgt, RGrp)`
> "If a more specific context already explains the occurrences of the sound correspondence, the more general context is less likely."

When seen without explanation, the rule "when subclass, then not superclass" might seem counterintuitive: After all, evidence for a subclass such as `rounded` is also evidence for its superclass (`vowel`). However, the formal interpretation of logical entailment does sometimes not correspond to human intuition, leading to naturally formulated PSL rules having unwanted consequences, as I will now illustrate.

Consider the following example: We have observed rounded and unrounded vowels, so `rounded = 1` and `unrounded = 1`. The desired context, obviously, is `vowel`. Now imagine the rules were phrased positively, i.e. "from subclass follows superclass":

```
1.0: Pcxt(LCls, Trgt, R) & Pdsb(LCls, LGrp) -> Pcxt(LGrp, Trgt, R)
1.0: Pcxt(L, Trgt, RCls) & Pdsb(RCls, RGrp) -> Pcxt(L, Trgt, RGrp)
```

Since the antecedent, `rounded` or `unrounded`, is `1`, the consequent, `vowel`, must also be `1`. Now the system considers all three classes equally likely. To prevent this, we have the `singleContext` rule, which will correctly select `vowel`. However, now that `vowel = 1`, there is pressure to apply the same rule with `vowel` in the antecedent, since we have `vowel (1) -> anything (0)`, which is logically false and thus maximizes the distance to satisfaction. The positive rules will therefore lead to `anything` always being the only acceptable solution.

With the negative rules, there is no such issue: Here, having `rounded = 1` and `unrounded = 1` will set `vowel = 0`. However, when `singleContext` comes into play, the belief of both

`rounded` and `unrounded` is lowered, and with it the negative pressure on `vowel`. Since `vowel` also receives the same positive evidence as both `rounded` and `unrounded` together via `contextObservation`, it will eventually be cheaper for the system to set `vowel = 1` and both subclasses to `0`. Since `0 -> 1` is a true entailment, `notSupergroupPcxt` poses no problem for this. On the other side, when there is only evidence for `rounded`, but not for `unrounded`, `notSupergroupPcxt` prevents the system from choosing `vowel` over its equally plausible subclass.

4.5 Phase 3: Sound Law Inference

The final phase, the actual sound law inference, combines the results from the previous two phases to infer the conditioned and unconditioned sound laws for all target languages, thereby completing step 4 of the comparative method (cf. section 1.3.1.4). Before starting this phase, the `Pcxt` atoms inferred previously are fixated and those with belief < 0.3 are deleted, since they can be considered rejected by the system and are usually nonsensical, so they should not have influence on the sound law inference.

4.5.1 Predicates

> **`Plaw`**`(ProtoLanguage, TargetLanguage, ProtoSound, TargetSound,`
> `LeftContext, RightContext)`
> "There was a sound change `ProtoSound` → `TargetSound` | `LeftContext _ RightContext` from `ProtoLanguage` into `TargetLanguage`."

The only new predicate in this phase is the open predicate `Plaw` which represents the inferred sound laws. Within a single SoInEn inference, the `ProtoLanguage` argument will always be the same, which is why it is hard-coded in the rules. At the current state of the system, it is useless, but since the long-term goal is to run SoInEn on several subfamilies at once, it has already been included.

The contexts can be any phoneme or sound class, `"anything"` if one or both sides are unconditional, or `"elsewhere"` if the sound law should apply to all contexts not covered by some conditional sound law. The conditioned Kannaḍa sound change `p > h / # _` , for instance, would be rendered as the following two atoms:

```
Plaw("pdrav", "kan", "p", "h", "#", "anything")
Plaw("pdrav", "kan", "p", "p", "elsewhere", "elsewhere")
```

The unconditioned Telugu law `ɭ > l`, on the other hand, would only require one atom, which already covers all possibilities:

```
Plaw("pdrav", "tel", "ɭ", "l", "anything", "anything")
```

4.5.2 Ideas

The algorithm for generating `Plaw` candidates is not particularly complicated, but deeply nested, which is why it is illustrated in Figure 4.15.

```
generateSoundLawCandidates():
    for each PcorID:
        C ← {(L, R) | Pcxt(L, PcorID, R) > 0.3}
        C ← C ∪ {(LSup, RSup) | L ⊆ LSup, R ⊆ RSup, (L, R) ∈ C}
        for Proto in {Proto | Prec(PcorID, Proto) > 0.5}:
            for (Lang, Sound) in PcorID:
                if Proto = Sound:
                    inject(Plaw("ProtoLang", Lang, Proto, Sound,
                            "elsewhere", "elsewhere"))
                else:
                    for (L, R) in C:
                        L ← convert(L)
                        R ← convert(R)
                        inject(Plaw("ProtoLang", Lang, Proto, Sound, L, R))

convert(Context):
    if Context = "unknown":
        return "anything"
    else:
        return Context
```

Figure 4.15: The algorithm for generating sound law candidates given the results of the first two inference phases.

We consider all proto-sounds reconstructed for a sound correspondence with belief > 0.5. For each member of the sound correspondence, we check whether it is equal to the proto-sound: If yes, no change has happened and we insert the previously mentioned default `Plaw` atom with `"elsewhere"` context. Otherwise, we fetch all contexts of this sound correspondence with belief > 0.3 and inject `Plaw` candidates for these contexts and for all contexts that consist of superclasses of these contexts. All `"unknown"` contexts are replaced with `"anything"`. The threshold for `Pcxt` atoms to be considered is comparatively small, because during inference, we will need as much evidence as possible from `Pcxt`s for different sound correspondences leading to the same sound changes in order to generalize them up to a single matching context, similar to how we did it with the `Pnei`s in the previous phase.

As an example, consider the following atoms:

```
Prec("h/p/p", "p") = 0.9
Pcxt("#", "h/p/p", "unknown") = 1.0
Pcxt("#", "h/p/p", "vowel") = 0.6
Pcxt("#", "h/p/p", "unrounded") = 0.4
Pcxt("#", "h/p/p", "rounded") = 0.2
Pcxt("unknown", "h/p/p", "vowel") = 0.4
```

Here, a change took place only in the first language. Hence, conditioned sound law candidates will be generated only for this language. Also, the belief of `Pcxt("#", "h/p/p", "rounded")` is too low for it to be included in the sound law contexts. Overall, the following `Plaw` atoms will be injected:

```
Plaw("ProtoLang", "Lang1", "p", "h", "#", "anything")
Plaw("ProtoLang", "Lang1", "p", "h", "anything", "anything")
Plaw("ProtoLang", "Lang1", "p", "h", "#", "vowel")
Plaw("ProtoLang", "Lang1", "p", "h", "#", "sonorant")
Plaw("ProtoLang", "Lang1", "p", "h", "anything", "vowel")
Plaw("ProtoLang", "Lang1", "p", "h", "anything", "sonorant")
Plaw("ProtoLang", "Lang1", "p", "h", "#", "unrounded")
Plaw("ProtoLang", "Lang1", "p", "h", "anything", "unrounded")
Plaw("ProtoLang", "Lang2", "p", "p", "elsewhere", "elsewhere")
Plaw("ProtoLang", "Lang3", "p", "p", "elsewhere", "elsewhere")
```

4.5.3 Rules

> ***levelPriorPlaw:***
> `0.5 / 5^(lvl - 1): Plvl(L, "lvl") & XPlaw("PLng", TLng, PSnd, TSnd, L, R) -> ~Plaw("PLng", TLng, PSnd, TSnd, L, R)`
> `0.5 / 5^(lvl - 1): Plvl(R, "lvl") & XPlaw("PLng", TLng, PSnd, TSnd, L, R) -> ~Plaw("PLng", TLng, PSnd, TSnd, L, R)`
> "The more general a context, the less likely it is by default."

This rule works just like `levelPriorPcxt` from the previous phase: It provides tiered negative priors for the sound class hierarchy.

> ***someSoundLawForEveryProto:***
> `Plaw("PLng", TLng, PSnd, +TSnd, +L, +R) >= 1 .`
> "There must be at least one sound change from a proto-sound into every target language."

Every sound in the proto-language's phoneme inventory must have some reflex in each daughter language, which is enforced by this rule. When there is no evidence for any conditioned sound change, this will promote the `"elsewhere"` sound law, since it has no negative prior.

> ***singleSoundLawContext:***
> `Plaw("PLng", TLng, PSnd, TSnd, +L, +R) <= 1 .`
> "A sound change can only apply in a single context."

Though technically not true, this rule is required to make PSL settle on one context per sound law and not assign them all high belief.

> ***elsewhere:***
> `1.0: Plaw("PLng", TLng, PSnd, TSnd1, "elsewhere", "elsewhere")`
> `= 1 - Plaw("PLng", TLng, PSnd, +TSnd2, "anything", "anything")`
> "There should be exactly one unconditioned sound change."

The `"elsewhere"` sound law as a "fallback" only makes sense when there are uncovered contexts. Hence, it is in direct competition with a sound change from the same proto-sound having a two-sided `"anything"` context. For example, when `Plaw("pdrav", "kan",`

`"ɻ", "ɭ", "anything", "anything"`) receives belief 1.0, i.e. the system correctly infers that Proto-Dravidian /ɻ/ became /ɭ/ everywhere in Kannaḍa, `Plaw("pdrav", "kan", "ɻ", "ɻ", "elsewhere", "elsewhere")` is not needed anymore, since all instances of proto /ɻ/ are already covered by the unconditioned sound change to /ɭ/.

> ***contextSupport:***
> `1.0: Pcxt("unknown", ID, RCtxt) - (1 - Prec(ID, PSnd)) <= Plaw("PLng", TLng, PSnd, +TSnd, +L, +R) {TSnd: Pcos(ID, TLng, TSnd)} {R: Psub(RCtxt, R)}`
> `1.0: Pcxt(LCtxt, ID, "unknown") - (1 - Prec(ID, PSnd)) <= Plaw("PLng", TLng, PSnd, +TSnd, +L, +R) {TSnd: Pcos(ID, TLng, TSnd)} {L: Psub(LCtxt, L)}`
> `1.0: Pcxt(LCtxt, ID, RCtxt) - (1 - Prec(ID, PSnd)) <= Plaw("PLng", TLng, PSnd, +TSnd, +L, +R) {TSnd: Pcos(ID, TLng, TSnd)} {R: Psub(RCtxt, R)} {L: Psub(LCtxt, L)}`
> "That a regular sound correspondence `ID` with `TSnd` as the `TLng` sound, for which `PSnd` was reconstructed, occurs between `LCtxt` and `RCtxt` indicates a sound change in this or a superclass context."

In this rule, the results from the previous two inference phases, proto-sound reconstruction and context detection, come together to provide support for conditioned sound laws. Subtracting the negated `Prec` belief from the `Pcxt` is a somewhat ad-hoc attempt to let the certainty of the reconstruction have some influence on the resulting sound laws. Multiplying the belief values of two atoms is unfortunately not possible in PSL.

Note that `TSnd` is not actually summed over, since there is only a single `TSnd` that matches `Pcos(ID, TLng, TSnd)` (i.e. that occurs in the sound correspondence `ID`), but we need the filter clause to ensure that the sound correspondence actually matches the sound laws, and only sum variables can be filtered.

> ***notSupergroupPlaw:***
> `1.0: Plaw("PLng", TLng, PSnd, TSnd, LCls, R) & Pdsb(LCls, LGrp) -> Plaw("PLng", TLng, PSnd, TSnd, LGrp, R)`
> `1.0: Plaw("PLng", TLng, PSnd, TSnd, L, RCls) & Pdsb(RCls, RGrp) -> ~Plaw("PLng", TLng, PSnd, TSnd, L, RGrp)`
> "If a more specific context already explains the occurrences of a sound change, the more general context is less likely."

This is the sound law variant of `notSupergroupPcxt`. Again, it serves to make the context as specific as possible and to keep PSL from overgeneralizing.

> ***noContextOverlap:***
> `1.0: Plaw("PLng", TLng, PSnd, TSnd1, L1, R) & XPlaw("PLng", TLng, PSnd, TSnd2, L2, R) & (TSnd1 != TSnd2) & (L1 != "elsewhere") & (L2 != "elsewhere") & Psub(L1, L2) -> ~Plaw("PLng", TLng, PSnd, TSnd2, L2, R)`
> `1.0: Plaw("PLng", TLng, PSnd, TSnd1, L, R1) & XPlaw("PLng", TLng,`

```
    PSnd, TSnd2, L, R2) & (TSnd1 != TSnd2) & (R1 != "elsewhere") &
    (R2 != "elsewhere") & Psub(R1, R2) -> ~Plaw("PLng", TLng, PSnd,
    TSnd2, L, R2)
```

"A proto-sound cannot become two different target sounds in the same language in overlapping contexts."

Finally, this rule ensures that conditioned sound changes for the same proto-sound and the same target language remain mutually exclusive, in that the same proto-sound cannot become two different sounds in overlapping contexts. Two contexts overlap when the left or right side of one sound law is the superclass of the respective side of the other.

5 Evaluation

In this chapter, I evaluate SoInEn's performance by comparing the proto-phoneme reconstructions and sound laws it infers for the Dravidian and Samoyedic languages in NorthEuraLex to the gold standard developed in chapter 3. I first introduce the general evaluation setup and the modifications I had to apply to the gold standard in section 5.1. Then I explain the different metrics I employ to measure SoInEn's success in comparison to the gold standard in section 5.2. Finally, I present and discuss the results of the evaluation in section 5.3.

5.1 Setup

To obtain the results presented in this chapter, I performed five consecutive runs of all three phases of SoInEn on both the Dravidian and Samoyedic data, and calculated the average of these results to account for any variation. The values were very stable though, with only minor differences between runs. The evaluation was performed on a machine with an Intel-Core i7-8750H 2.2 GHz CPU with 4 GB of RAM provided to the JVM, though they were never completely used over the course of the entire five runs.

While the current published version of NorthEuraLex is 0.9, work on the forthcoming version 0.92 has progressed quite far already, and it comes with improved transcription for many languages, notably Dravidian and Samoyedic. Since SoInEn can benefit from this, I have supplied it with the NorthEuraLex 0.92 transcriptions. In addition, because the Samoyedic cognates are notoriously hard to detect automatically, I provided SoInEn with gold standard cognacy annotations that were extracted from Janhunen's (1977) etymological dictionary by Anna Bródy in the context of the EtInEn project.

5.1.1 Format of the Evaluation Files

In order to be able to evaluate SoInEn's output against them, the gold standard sound laws for Dravidian and Samoyedic compiled in chapter 3 need to be stored in a machine-readable format. Since SoInEn operates on non-cluster targets and contexts only and

```
#groups
long  i: e: a: o: u:

#inventory
i i: e e: a a: o o: u u: p t̪ t ʈ t͡ɕ k ...

#soundlaws
//lang  proto target  left  right set  comment
kan,tel  i  e  0  *a    with /a/ in next syllable
kan  p  h  #  0
kan,tel  p: p  0  long
tam,mal  mp:  p: 0  0
kan,tel  mp:  mp 0  0
```

Figure 5.1: Excerpt from the basic evaluation file for Dravidian, with tab characters explicitly marked to show the format.

NorthEuraLex's phonetic transcription of the data differs slightly from what is assumed in the literature in several places, some of the sound laws have to be refactored to give SoInEn a chance to actually find them.

The gold standard sound laws I derived can also be useful as a test case for other computational linguists working on automated sound law inference. Apart from the gold standard refactored to fit the NorthEuraLex data, I thus also provide a basic version closely matching what I discussed in chapter 3. An excerpt of that file for Dravidian is shown in Figure 5.1 to illustrate the file format.

The gold standard file is separated into three parts: The first (optional) part is introduced by the line `#groups` and allows the user to specify "sound classes" apart from those compiled for SoInEn (cf. section 4.2.5) in the format `<group name>`, tab, members separated by whitespaces. Then, after the line `#inventory`, the proto-phoneme inventory is enumerated, with all sounds on one line, separated by whitespaces. Finally, the line `#soundlaws` starts the actual sound law section, which is written in a tabular format, with the seven columns specifying the target language(s), the proto-sound, the target sound, the left and right context, the sound law set (to be explained later), and some comment on the respective sound law. Empty ("anything") contexts can be specified by explicitly writing `anything` or its shorthand `0`, or by leaving the cell empty. A context starting with an asterisk `*` is also interpreted as an empty context, but can be used to distinguish it from other empty contexts in sound correspondence generation (see section 5.1.2). At any point in the file, a line can be marked as a comment by starting it with `//`.

The above file, however, cannot be direct input to SoInEn, because it partially operates on consonant clusters, and because SoInEn only knows short vowels and single consonants (with a long vowel or geminate consonant being interpreted as sequences of two short vowels/consonants). The refactoring of the Dravidian cluster rules is not trivial, and some of the contexts cannot be represented accurately (or at all). Consider, for instance, the Kannaḍa and Telugu rule p: > p / _ [long]: Since the proto-sound cannot be a geminate,

```
kan    i    ɛ    0    *a    with /a/ in next syllable
tel    i    e    0    *a    with /a/ in next syllable
kan    p    h    #    0
kan    p    -    p    0    kan-pp
kan    p    -    0    p    kan-pp
tel    p    -    p    0    tel-pp
tel    p    -    0    p    tel-pp
tam,mal    m    -    0    p
```

Figure 5.2: The sound laws from the file in Figure 5.1, with clusters resolved and adapted to the NorthEuraLex data.

we could instead have a sound law p > ∅ / p _ [long]. However, SoInEn might end up deleting the first [p] instead, inferring a sound law p > ∅ / _ p and leaving the [p] before the long vowel untouched. Hence, according to our gold standard, either of the two should be fine. This is where the "sets" column of the file format comes into play: When specified, finding one member of a set is sufficient for fulfilling the gold standard for the whole set. We can therefore create a set kan-pp containing both p > ∅ / p _ [long] and p > ∅ / _ p. One problem remains: The long vowel context cannot be matched by SoInEn. We could replace it by a single vowel context (which would be incorrect though) or leave it unspecified.

Finally, the phonetic transcription used in NorthEuraLex is not phonemic, for instance rendering phonological processes like assimilation or final devoicing applying in individual languages. There might also be variation in how the same phoneme is rendered in different languages: Kannaḍa /e/, for instance, is only transcribed [e] word-finally, but [ɛ] elsewhere, corresponding to how it is actually pronounced. The gold standard sound law i > e therefore has to be rendered i > ɛ for Kannaḍa, since it never applies word-finally. This becomes more complicated when the sound is rendered differently in several languages in varying contexts. Short Dravidian /a/, for example, is (in version 0.92!) transcribed [ə] in non-initial syllables in Tamiḻ and Malayāḷaṁ, [ə] except word-finally in Kannaḍa, and [ʌ] everywhere in Telugu. Since it is most commonly rendered in a reduced form in all daughter languages, it could and probably will be reconstructed as [ə] instead of [a], which I would not consider an error. I therefore decided to add [ə] to the proto-phoneme inventory for Dravidian.

Figure 5.2 shows the final refactoring of the sound laws from Figure 5.1. The result is not quite satisfying for the consonant clusters, where crucial distinctive information was lost due to the restricted format. It is to be expected that SoInEn performs poorly at detecting these sound laws, since their actual contexts cannot even be rendered as a `Plaw` atom. There are similarly difficult cases for Samoyedic: Diphthongs, for instance, are also considered sequences of two separate vowels inside SoInEn. Rendering the diverse sound changes of the Samoyedic diphthongs inside the `Plaw` paradigm is a hopeless endeavor, and so they were omitted from the final gold standard. The NorthEuraLex transcription also proved to be problematic for detecting some of the sound laws. Most drastically, the Enets data did not represent the glottal stop in any way, so all sound laws producing it in

Enets had to be rewritten as deletions.

In the end, the Dravidian gold standard contains 28 proto-phonemes and 185 sound laws, while the Samoyedic gold standard contains 22 proto-phonemes and 192 sound laws, rendering them of comparable size. The complete refactored gold standard files can be reviewed in Appendix C.

5.1.2 Generating Gold Standard Sound Correspondences

Sound correspondences with their proto-sound reconstructions are not directly specified in the gold standard, but instead derived from the sound laws. Initially, one default sound correspondence is created for each sound in the gold standard proto-inventory from the unconditioned sound laws, with remaining gaps filled in by the proto-sound itself. Then, one sound correspondence is created for each proto-sound and distinct context in which a sound law applies to this proto-sound, with the gaps filled in from the default sound correspondence created previously.

The sound laws from Figure 5.2, for example, would first yield the default sound correspondences i/i/i/i and i/i/ɛ/e (in the starred "anything" context `*a`) for proto *[i], p/p/p/p for *[p] and m/m/m/m for *[m]. Then, the conditioned sound laws are applied to additionally derive the regular sound correspondences p/p/h/p (word-initially) and p/p/-/- (before or after *[p]) for *[p] as well as -/-/m/m (before *[p]) for *[m].

As we can see, the starred "anything" contexts are treated as different than plain `anything`, making it possible to create several sound correspondences for the same context. This is useful, for instance, when a sound law applies in a context that is not directly specifiable as a single adjacent sound, but will still generate a distinct sound correspondence, as for the Dravidian vowel changes when there is short *[a] in the following syllable: The context itself cannot be specified, but there will be two different sound correspondences for each of the affected proto-vowels (like i/i/ɛ/e and i/i/i/i for *[i]), one in the context of short *[a] in the following syllable and one occurring elsewhere.

5.2 Method

In this chapter, I only present and discuss the evaluation results of phase 1 (proto-inventory reconstruction) and 3 (sound law inference). I omit the evaluation of phase 2 (context detection) for two reasons: First, in contrast to proto-sound reconstructions and sound laws, contextual occurrences of sound correspondences would not be considered an outcome of the comparative method by a historical linguist. The `Pcxt` atoms in SoInEn merely function as aggregators of information, as an intermediate step between sound correspondences, reconstructions and sound laws. Second, `Pcxt` is particularly hard to reasonably evaluate, since both the actually found sound correspondences and contexts might deviate from what is demanded in the gold standard, leading to a combinatory problem.

Consider, for instance, the extremely regular Kannaḍa sound law `ʊ > b / # _` which creates the sound correspondence ʊ/ʊ/b/ʊ for the four Dravidian languages. Kannaḍa [b], however, can also be observed where some of the other have [ʊ] inside a word in the

NorthEuraLex data, so instead of `Pcxt(#, ʊ/ʊ/b/ʊ, anything)`, the model generates and assigns high belief to e.g. `Pcxt(anything, ʊ/ʊ/b/ʊ, sonorant)`, which is also an acceptable context for the sound correspondence ʊ/ʊ/b/ʊ. On the other side, we have, for instance, the Tamil sound law `g > ɣ / V _ V`, which should yield the atom `Pcxt(vowel, ɣ/g/g/g, vowel)`. However, ɣ/g/g/g is observed so rarely that no `Pcxt` atom is generated for it at all, which is a general problem with those sound laws applying in less frequent contexts. Also, *[g] is sometimes rendered [k] in one of the other languages, yielding sound correspondences like ɣ/g/g/k, ɣ/g/k/k or ɣ/k/k/k. The latter is actually frequent enough to receive a `Pcxt` atom, namely `Pcxt(vowel, ɣ/k/k/k, unknown)`, which should in some way count for the gold standard `Pcxt(vowel, ɣ/g/g/g, vowel)` because it correctly implies that Tamil develops [ɣ] next to a vowel.

Both loosely matching context and sound correspondence, however, is extremely difficult and probably impossible to do automatically with reasonable results. We would not, for instance, want something like `Pcxt(unknown, p/p/b/p, anything)` to count for the gold standard `Pcxt(#, ʊ/ʊ/b/ʊ, anything)`, even though the distance between the two is similar to that between `Pcxt(vowel, ɣ/k/k/k, unknown)` and `Pcxt(vowel, ɣ/g/g/g, vowel)`. Since `Pcxt` atoms are not targets according to the comparative method anyway, I have decided not to evaluate them.

5.2.1 General Measures

For each of the remaining target predicates (`Pinv`, `Prec` and `Plaw`), I calculate both precision p and recall r during the phase in which they are introduced. However, "pure" precision and recall are not so meaningful when it comes to PSL atoms, since they do not incorporate the atoms' belief values: If all gold standard atoms were generated, but assigned belief 0.0 by the model, the recall would amount to 100%, though we would actually like it to be 0% since a 0.0 belief atom is equivalent to the non-existence of that atom. Similarly, if our gold standard contained 10 atoms, which were all generated and assigned belief 1.0, but the model also generated 100 garbage atoms with belief 0.0, the precision would amount to only 9%, even though the model correctly distinguished the gold standard from the undesirable atoms. I therefore normalize the true positive (n_t) and false positive counts (n_f) by the average true positive (b_t) and false positive belief (b_f) and calculate *normalized* precision p' and recall r' as follows:

$$p' = \frac{b_t \cdot n_t}{b_t \cdot n_t + b_f \cdot n_f} \tag{5.1}$$

$$r' = b_t \cdot r \tag{5.2}$$

In this way, the lower an atom's belief, the more it is counted like a negative, which reflects the semantics of belief values.

In addition, the average belief values of true positives and false positives are also provided explicitly as an indicator of how well the model distinguishes between "good" and "bad" ideas. However, it is often the case that much more "bad" ideas are generated than "good" ones, which makes the false positive belief less representative. There are, for instance, 69

	Dravidian		Samoyedic	
	Target/Total Atoms	**Runtime**	**Target/Total Atoms**	**Runtime**
Phase 1	2,992/33,306	7.193 sec	2,232/22,884	5.859 sec
Phase 2	7,376/48,486	6.362 sec	731/22,204	2.317 sec
Phase 3	379/34,781	6.352 sec	123/21,078	2.322 sec

Table 5.1: General performance of SoInEn in terms of generated atoms and runtime for the three different inference phases.

`Prec` atoms in the Dravidian gold standard, not all of which are actually found, and almost 2800 additional `Prec` atoms generated that are not in the gold standard. Since by design, SoInEn can only assign a small fraction of these 2800 atoms high belief, the average false positive belief will always be low, even though there could be 100 "bad" atoms with 1.0 belief.

To account for this, I also calculate the average belief of the n highest rated false positives, where n is the number of generated gold standard atoms. So if, for example, 40 gold standard `Prec` atoms are found during evaluation, the 40 non-gold standard atoms with the highest belief are picked and their average belief is calculated to get a better idea of whether the gold standard atoms are actually higher rated than the false positives.

5.2.2 Loose Context Matching

The issue of alternative acceptable contexts that made `Pcxt` evaluation difficult also persists for `Plaw` atoms. To account for this, I evaluate all conditioned `Plaw`s twice: First, only those `Plaw` atoms *exactly matching* the gold standard contexts are counted. Then, for a *loosely matching* evaluation, all `Plaw` atoms whose contexts are equal to, a subclass of, or a superclass of the gold standard context are considered, and the one with highest belief is picked as the one matching the gold standard. For the gold standard atom `Plaw(pdrav, kan, p, h, #, anything)`, for example, `Plaw(pdrav, kan, p, h, anything, anything)`, `Plaw(pdrav, kan, p, h, #, vowel)` or `Plaw(pdrav, kan, p, h, #, alveolar)` would also be accepted as loosely matching the required gold standard `Plaw`.

5.3 Results and Discussion

In general, SoInEn is better able to find useful patterns in the Dravidian than in the Samoyedic data, as evidenced by the much higher number of generated (target) atoms shown in Table 5.1 despite the similarly large gold standard. This imbalance is especially strong when it comes to context detection, where there are ten times more target atoms generated for Dravidian than for Samoyedic, indicating that SoInEn has trouble deriving regular conditional occurrences of sound correspondences from the Samoyedic input data. The lack of context propagates into sound law inference, where Dravidian spawns thrice as many target atoms as Samoyedic, which ends up having less candidate atoms than sound laws in its gold standard. I investigate the quality of the produced target atoms of phases 1 and 3 in more detail in the following sections.

	Dravidian		Samoyedic	
	Pinv	**Prec**	**Pinv**	**Prec**
Precision	38.89%	1.59%	42.97%	0.90%
Norm. Precision	83.99%	29.08%	75.91%	16.58%
Recall	100.00%	62.70%	91.67%	27.22%
Norm. Recall	81.21%	42.55%	53.54%	15.21%
Recall*		97.89%		73.68%
Norm. Recall*		66.43%		41.18%
Found GS Belief	0.812	0.679	0.584	0.559
Non-GS Belief	0.099	0.027	0.140	0.025
Top n Non-GS Belief	0.155	0.896	0.185	0.863
Alt. Rec. Belief		0.071		0.064

Table 5.2: Evaluation results of phase 1. "(Norm.) Recall*" refers to the recall when only GS proto-reconstructions for sound correspondences actually occurring in the data are required. "Alt. Rec. Belief" is the average belief of `Prec` atoms encoding non-GS reconstructions for GS sound correspondences.

Since it has fewer atoms to optimize for Samoyedic, SoInEn's inference runs considerably faster on Samoyedic than on Dravidian, though it is quite fast on both families considering the high complexity of the inference problem.

5.3.1 Phase 1: Proto Inventory Reconstruction

The evaluation results for `Pinv` and `Prec`[1], the target atoms for SoInEn's inference phase 1, are summarized in Table 5.2.

Proto inventory reconstruction works quite well for Dravidian, as evidenced by the high normalized precision and recall and the distinctive belief values for GS and non-GS `Pinv` atoms. It generates all 28 gold standard proto-phonemes as candidates and assigns high belief to all except [t], [d], [l̪] and [ɲ]. This is to be expected for [t] and [d], since they have almost no reflexes in the daughter languages, and for [ɲ] which was rare already in Proto-Dravidian and lost in Kannaḍa and Telugu. [t͡ʃ] and [d͡ʒ] are assigned belief values around 0.6 and 0.5, respectively, so that [d͡ʒ] is always and [t͡ʃ] sometimes deleted before the next phase because their belief is too low. SoInEn falsely assigns high (> 0.6) belief to [r], [s] and [h], which is acceptable, because [r] occurs in all daughter languages and is the regular reflex of the missing [t], and [s] and [h] are both common reflexes of the often-missing [t͡ʃ] as well as frequent in loanwords.

For Samoyedic, normalized precision and recall as well as the average GS belief is much lower. While 22 of 24 gold standard atoms are generated, SoInEn assigns low belief to many of them. It performs especially poor on the vowels, where it only manages to reconstruct [i], [e] and [o], plus [ɑ] instead of GS [ɒ], with high belief, assigning beliefs of < 0.4 to the other 7 GS vowels. It also fails to assign high belief to the unstable glides [w]

[1] As a convention, when citing sound correspondences in this the following section, the order of sounds will be Tamil̲/Malayāḷaṁ/Kannaḍa/Telugu for Dravidian, and Nganasan/Enets/Nenets/Selkup for Samoyedic.

and [j], and [ŋ], instead reconstructing [g] (but not [b] or [d]) and [ʎ] (a common rendition of palatalized [l] in NorthEuraLex) as having been part of the Samoyedic proto-phoneme inventory. As to be expected, [t͡ʂ] is not even generated as a candidate.

The actual proto-phoneme reconstruction for sound correspondences, represented by the `Prec` atoms, seems to perform poorly for both language families. Precision, both raw and normalized, is quite low, since many `Prec` atoms not derivable from the gold standard sound laws are generated and assigned high belief. When comparing the found GS atoms to the same number of top-rated non-GS atoms, the model even seems to prefer the non-GS atoms ("Top n Non-GS Belief"). Recall is rather low as well. However, this is mainly because many of the sound correspondences for which `Prec` specifies the proto-phonemes are not even found in the data: When requiring only `Prec`s for those sound correspondences for which SoInEn has introduced a `Pcor` atom, recall improves considerably, and SoInEn only fails to generate those GS `Prec` atoms where the proto-sound has not survived in any of the descendant lexemes. Also, the average belief of alternative (non-GS) proto-phoneme reconstructions for these sound correspondences is extremely low, showing that SoInEn is indeed able to accurately distinguish good from bad proto-phoneme reconstructions when presented with a sound correspondence. This indicates that the issue is not the reconstruction process itself, but rather a combination of sparse data and poor sound correspondence detection.

Indeed, when inspecting the missing GS atoms, it turns out that many of the involved sound correspondences only occur in very specific contexts, making them extremely rare to observe. Samoyedic, for instance, lacks almost all of the sound correspondences where a consonant has been deleted[2] syllable-finally in Enets, such as b/-/p/p, ʔ/-/ʔ/k, s/-/s/s or r/-/r/r (with the exception of ʔ/-/ʔ/t). Since consonant clusters, especially of obstruents, are uncommon in Samoyedic, these sound correspondences cannot be observed very frequently (if at all), and are probably interpreted as noise by SoInEn.

This issue is amplified by the general data sparsity, especially for Samoyedic, for which there are only 536 cognate sets comprising more than one language in NorthEuraLex, even with the gold standard cognacy annotation. 415 (or 77.43%) of these contain lexical items from only 2 languages, 108 (or 20.15%) from 3 languages, and only 13 (or 2.43%) have representatives for all 4 languages. It is virtually impossible to find several occurrences of 73 gold standard sound correspondences among these few cognate sets. For Dravidian, which also has 73 gold standard sound correspondences, the data coverage is a little better, which is also reflected in the higher recall: Of 705 total cognate sets, 470 (or 66.67%) have data for only 2 languages, 163 (or 23.12%) for 3 languages, and 72 (or 10.21%) have full coverage for all 4 languages. Not only are there more cognate sets overall in the Dravidian data, they are also larger, providing clearer evidence for sound correspondences.

What further contributes to the bad sound correspondence detection is the quality of the automated multiple sequence alignments, the difficulties with which I have already discussed in section 4.2.2. The algorithm usually functions well on the Dravidian data, because the lexical items within one cognate set are often extremely similar to each other.

[2] ...or turned into a glottal stop, which is not represented in the NorthEuraLex data, so it also looks like deletion there.

nio	biːʔ	–	–	b	i	i	ʔ	bɐc͡çedʲɐ	b	ɐ	c͡ç	e	–	d	ʲ	ɐ
enf	biu	b	i	u	–	–	–	bʲedɨ	b	–	ʲ	e	–	d	–	ɨ
yrk	juʔ	j	–	u	–	–	ʔ	jindʔ	j	–	–	i	n	d	–	ʔ
sel	køt	k	–	ø	–	–	t	kæjɨ	–	–	k	æ	–	–	j	ɨ

Figure 5.3: Faulty alignment for the Nganasan (nio), Enets (enf), Nenets (yrk) and Selkup (sel) words for 'ten' (left) and 'breath' (right) (Dellert and Jäger 2017).

It shows severe difficulties with aligning some of the more obscure Samoyedic cognate sets, which prevents detection of especially the more unusual sound correspondences. Consider, for instance, the alignments in Figure 5.3 produced for 2 of the only 13 full cognate sets in the Samoyedic data: They both properly reflect the various sound changes that apply to Proto-Samoyedic *[w], which became [b] anywhere in Nganasan, [b] word-initially in Enets, [j] before front vowels in Nenets and [k] word-initially in Selkup. However, since the alignment algorithm fails to correctly align the initial consonants in both cases, they do not contribute any evidence for the sound correspondence b/b/j/k at all.

The sound correspondence b/b/j/k exemplifies an additional problem that is caused by the evaluation method: Since the gold standard only explicitly lists the proto-phoneme inventory and the sound laws, the gold standard sound correspondences with their reconstructions are indirectly derived via the procedure described in section 5.1.2. Based on the gold standard sound laws for Samoyedic, this method demands a `Prec(b/b/w/k, w)` (word-initially) and a `Prec(b/w/j/w, w)` (before front vowels), but it does not merge these two into a `Prec(b/b/j/k, w)` (word-initially before front vowels). Proto reconstructions for the sound correspondence b/b/j/k will therefore always be counted as non-GS atoms, even though they are actually desirable.

Finally, some of the sound correspondences have GS proto-phoneme reconstructions that SoInEn simply cannot derive with the PSL rules it was given for proto-reconstruction, for instance *[d] for Dravidian r/r/r/ɽ and ɳ/n̪/d̪/ɖ, *[ɯ] for Samoyedic i/i/i/i, or *[t͡s̪] for Samoyedic ð/d/d/t and s/t/t/t, because the desired proto-phoneme is in some sound class that it shares with none of the phonemes participating in the respective sound correspondence.

Overall, given the sparse input data and often very noisy alignments, SoInEn's proto-reconstruction is much better than it looks by the raw numbers. For those sound correspondences that actually can be derived from the data, it usually comes up with sensible proto-phoneme reconstructions, and assigns low belief to the false alternatives. For Dravidian, proto-reconstructions works well enough to derive almost the complete gold standard proto-phoneme inventory. For Samoyedic, the reconstructed consonant inventory is also rather close to the gold standard, while the recall for its extremely unstable vowels, having undergone major changes in all daughter languages, is quite low.

5.3.2 Phase 3: Sound Law Inference

The results for the sound laws inferred in phase 3, the most difficult targets of SoInEn due to their high dependency on previous results, are summarized in Table 5.3.

	Dravidian				
	Plaw	**Plaw+**	**ID Plaw**	**NID Plaw**	**NID Plaw+**
Precision	23.58%	25.93%	75.27%	6.37%	8.50%
Norm. Precision	50.14%	61.28%	77.77%	4.18%	33.48%
Recall	47.46%	49.95%	79.55%	18.35%	23.09%
Norm. Recall	38.24%	46.51%	77.88%	2.28%	18.05%
Recall*	66.52%	70.32%		29.19%	36.86%
Norm. Recall*	53.60%	65.48%		3.63%	28.81%
Found GS Belief	0.806	0.931	0.979	0.125	0.782
Non-GS Belief	0.247	0.206	0.852	0.194	0.144
Top n Non-GS Belief	0.781	0.588	0.852	1.000	0.972

	Samoyedic				
	Plaw	**Plaw+**	**ID Plaw**	**NID Plaw**	**NID Plaw+**
Precision	37.87%	42.95%	80.99%	1.56%	10.42%
Norm. Precision	59.50%	63.51%	80.52%	0.00%	15.35%
Recall	23.44%	26.35%	69.84%	0.78%	5.12%
Norm. Recall	22.25%	23.75%	67.82%	0.00%	2.23%
Recall*	42.86%	48.19%		1.69%	11.19%
Norm. Recall*	40.69%	43.43%		0.00%	4.88%
Found GS Belief	0.949	0.901	0.971	0.000	0.438
Non-GS Belief	0.395	0.391	1.000	0.296	0.280
Top n Non-GS Belief	0.647	0.520	1.000	1.000	1.000

Table 5.3: Evaluation results of phase 3. "(Norm.) Recall*" refers to the recall when only GS sound laws for proto-phonemes successfully reconstructed in phase 1 are required. `Plaw+` are loosely matched atoms. `ID` refers to identity sound laws (with "elsewhere" context), `NID` (non-identity) to actual sound changes.

Considering the noisy input data and mediocre results of the previous phase, the relatively high precision of raw `Plaw` for both language families seems striking. As expected, given the difficulties faced in phase 1, recall is much higher for Dravidian than for Samoyedic, even when only requiring the sound laws for proto-phonemes previously reconstructed ("(Norm.) Recall*"). At least for Dravidian though, more than a third of the gold standard sound laws are found despite the poor input. However, the vast majority of these are identity sound laws (where the proto-sound does not change) with "elsewhere" contexts, i.e. the fallback default atoms that generally receive high belief by design when no conflicting unconditioned sound law is detected (`ID Plaw`). Real sound changes (`NID Plaw`) are only rarely found, and virtually never assigned high belief with the exact gold standard contexts. Also, there are always sound laws detected with very high belief that are not in the gold standard at all ("Top n Non-GS Belief").

5.3.2.1 Correctly Detected Sound Changes

For Dravidian, SoInEn detects most of the more frequently occuring sound changes, though usually assigns them more specific contexts than they have in the gold standard, probably due to their still limited number of occurrences. Examples of this are:

- Tamil̲ ɖ̪ > ð / V _ [close], belief 0.639 (GS: V _)
- Kannaḍa p > h / # _ V, belief 1.0 (GS: # _)
- Kannaḍa ʋ > b / _ [sonorant], belief 1.0 (GS: # _)
- Kannaḍa ɻ > ɭ / V _ [close], belief 0.558 (GS: anywhere)
- Telugu ɳ > n̪ / [sonorant] _ [sonorant], belief 1.0 (GS: anywhere)
- Telugu ɻ > ɖ / [unrounded] _ , belief 0.567 (GS: anywhere)

Most of these inferred contexts reflect Dravidian phonotactics: Since there are no consonant clusters except word-medial nasal+obstruent, SoInEn correctly detects that many of these sound changes only take place next to vowels (or sonorants, including adjacent nasals and approximants). Sometimes, as in the case of the Kannaḍa and Telugu laws for *[ɻ], the sound correspondences are only found next to a rather specific class of vowels due to their general infrequency. For the Kannaḍa sound law ʋ > b, SoInEn also overspecifies the right context, but overgeneralizes the left one. Looking at the evidence it has for this, we can see that there are two alignments where Kannaḍa [b] is aligned with [ʋ] in the other languages word-medially: Kannaḍa [kɔbːu] with Telugu [krouːu] 'fat' (cognacy to Tamil̲ [koɻʉpːʉ] and Malayāḷaṁ [koɻupːɨ] not detected), and Kannaḍa [ɕɪlʊbe] with Tamil̲ [t͡ʃilʉʋaɪ] and Telugu [s̪iluʋʌ] 'cross'. While the latter is a loanword from Syriac *šlibo* (Kittel 1894), the Kannaḍa cluster [rʋ] regularly becomes [bː] (Andronov 2003, p. 56), a sound change that was not included in the gold standard due to its limited occurrence and because it requires the cluster [rʋ] to arise via several other sound changes in the first place. The alternative contexts SoInEn infers for the Dravidian gold standard sound laws therefore make sense overall.

For Samoyedic, SoInEn unfortunately only detects three of the gold standard sound changes (plus four more to which it assigns 0 belief), namely:

- Nganasan p > h / # _ V, belief 0.865 (GS: _ V)
- Nenets k > x / # _ , belief 1.0 (GS: _ [back])
- Selkup k > q / # _ , belief 1.0 (GS: _ [open, close-mid])

For all three, there is a lot of evidence in the data, but almost all of the occurrences are at the beginning of a word. The Nganasan sound change p > h, for instance, occurs word-initially in 17 cognate sets, and word-medially only in one, namely the one for the concept 'fur' (Nganasan [kuhu] and Selkup [qopɨ]). The sound changes from *[k] suffer from a similar imbalance. In addition, SoInEn is unable to detect their right contexts because the sound correspondences to be found there are so diverse, to the extent that the 13 (partial) instances of word-initial k/k/x/q in the data occur left of 12 distinct vowel correspondences (e.g. o/ɑ/?/æ, i/i/ɨ/?, u/u/u/?, or ə/?/æ/ø). Given that only four proto-vowels could be reconstructed with high belief, it is clear that not all of these 12 different

vowel correspondence are frequent enough to have a proto-sound assigned. In fact, only one of them participates in a `Prec` with a belief higher than 0.5, the threshold for being included in `Pcxt` candidate generation. And even if they all had proto-sound reconstructions, some of the sound correspondences would be interpreted by SoInEn to have derived from non-back vowels, because all of the reflexes are non-back, as in Nganasan [kɐm], Nenets [xeːmʔ], Selkup [kæm] 'blood' (from Proto-Samoyedic **kɐ̨m*, Janhunen 1977). The contextual evidence is therefore just too diverse for SoInEn to derive the correct contexts for the above sound laws.

5.3.2.2 Missing Sound Changes

When a proto-sound was not reconstructed during phase 1, SoInEn will never assume any sound law for it. This accounts for many of the missing gold standard sound laws in both families, especially most of the vowel changes in Samoyedic, as evidenced by the sharp increase of recall when demanding only sound laws for proto-sounds previously inferred ("(Norm.) Recall*"). I will therefore only discuss the sound laws missing for correctly detected proto-phonemes.

We have already seen that the three Samoyedic sound laws detected by SoInEn all applied word-initially. Indeed, since Samoyedic consonants are extremely stable at the beginning of a word, the only word-initial sound change SoInEn missed is the palatalization of [k] before front vowels, for which the proto-reconstruction is difficult to get right since the sound change applies in all four languages. It is also so rare in the data that the only sound correspondence pertaining to *[k]-palatalization is s/s/s/q, which is traced back to proto *[s] with a belief of only < 0.3. So while the word-initial sound laws are reasonably well detected, SoInEn does not find any of the word-medial changes, namely the ones applying intervocally and the consonant deletions (Enets) or glottal stop replacements (Nganasan, Nenets) in coda position. As already discussed in the previous section, these occur in such specific contexts that SoInEn is even unable to detect the respective sound correspondences. All `Prec` atoms containing Nganasan or Nenets [ʔ], for instance, receive a belief of < 0.01, rendering the generation of any such sound change impossible.

For Dravidian, the reasons for missing gold standard sound changes are a little different. First of all, as already mentioned in section 5.1.1, many of the Dravidian sound laws are difficult to model in the restricted syntax of SoInEn's `Plaw` atoms. Most of them apply to geminates or nasal+obstruent clusters, a context that cannot be captured properly with `Plaw`s that can only apply to a single sound with one neighboring sound or sound class on each side. Also, these contexts are again very specific and will likely not all occur in the data. As a consequence, none of the consonant cluster sound laws could be identified by SoInEn. Then there are also several vowel changes that are conditioned on the vowel of the following syllable, which cannot even be approximately modeled as a `Plaw`. They are actually frequent enough for SoInEn to detect some of them, namely Malayāḷaṁ o > u / [occlusive] _ [occlusive], Kannaḍa i > ɛ / _ [voiced] and Telugu i > e, as well as Tamiḻ u > o / C _ for GS o > u and Telugu o > u / _ C for GS u > o (all five actually applying with *[a] in following syllable). Still, the latter two are technically not correct, and several more of these are still missing.

Finally, Dravidian *[k]-palatalization suffers from the same problem as the respective Samoyedic sound laws: The change k > t͡ɕ|t͡ʃ / # _ [front] happens in all four languages except Kannaḍa. In addition, NorthEuraLex contains only one cognate set in which it applies that has a Kannaḍa lexeme, namely Malayāḷaṁ [t͡ɕɛʋi], Kannaḍa [kɪʋi], Telugu [t͡ɕɛʋi] 'ear', which is not enough evidence for giving any reconstruction for t͡ʃ/t͡ɕ/k/t͡ɕ a high belief (*[t͡ʃ] receives belief 0.013, *[k] plain 0.0).

5.3.2.3 Falsely Detected Sound Changes

As indicated by the "Top n Non-GS Belief", SoInEn detects several false sound laws with very high belief. Note that because the "Top n Non-GS Belief" is supposed to provide a comparable value to the "Found GS Belief", the number of atoms averaged for it correspond to the number of found gold standard atoms, which often is quite low, especially for Samoyedic. Therefore, while it seems extreme that the "Top n Non-GS Belief" is 1.0 for Samoyedic `NID Plaw`, remember that this is only the average belief of the 1 (exactly matched) and 6 (loosely matched) highest rated non-GS atom(s).

For the identity sound laws, the false positives are those referring to the falsely inferred proto-sounds, as well as those for proto-sounds that underwent unconditioned sound changes in the gold standard but were detected in more specific contexts by SoInEn, making the fallback identity law necessary. Since "elsewhere" `Plaw`s get high belief by default, the high average non-GS belief for false identity sound laws is not particularly surprising.

There are several false non-identity sound changes inferred by SoInEn for which there is only little direct evidence in the data, which however gets blown up by combining it with other compatible partial alignments. We find, for instance, a Tamil̲ sound law h > ɣ / _ [sonorant]. [h] is not in the gold standard inventory of Proto-Dravidian, and indeed, the only actual evidence for this are loanwords deriving from Sanskrit *guhā* 'cave' (cf. Monier-Williams 1899), namely Tamil̲ [kuɣaɪ], Malayāḷaṁ [gufia], Kannaḍa [gʊhe] and Telugu [guhʌ]. However, there are many other alignments of Malayāḷaṁ [fi] with Kannaḍa and Telugu [h], not containing a Tamil̲ cognate, which are counted as evidence for a regular sound correspondence ɣ/fi/h/h. To prevent something like this from happening, the Gapped Frequency Trie performing these combinations (cf. section 4.2.3.1) is only allowed to merge two (partial) sound correspondences when they share at least two different symbols, but this condition is unfortunately satisfied by the [fi]∼[h] contrast. In the same way, evidence is "generated" for several other false sound laws, for example:

- Tamil̲ s > w / _ [sonorant] via one observation of w/s/s/s̱ combined with several observations of ?/s/s/s̱
- Telugu i > ʌ / C _ C via one observation of i/i/ɪ/ʌ combined with several observations of i/i/ɪ/?
- Nganasan g > k / # _ and Selkup g > q / # _ via one observation of k/g/?/q combined with several observations of ?/?/g/q and k/?/?/q (more likely an instance of k > g in Enets and Nenets)

Other non-GS sound laws with high belief are actually justified in light of the data. When

discussing the difficulties with `Prec` evaluation in section 5.2, I already mentioned that the Dravidian gold standard sound correspondence ɣ/g/g/g is sometimes found as ɣ/k/k/k. This makes SoInEn infer a sound law k > ɣ / V _ for Tamil̲ instead of GS g > ɣ / V _ with belief 1.0, which is quite good given the noisy input data. Similarly, it finds Nganasan t > c͡ç / # _ and Nenets n > ɲ / _ , which can both be counted toward GS ∅ > ʲ / C _ [front] given that [c͡ç] and [ɲ] are NorthEuraLex transcriptions of underlying /tʲ/ and /nʲ/, respectively. Then there are regular patterns in the data that are not part of the gold standard: There are several Tamil̲ words for the form CVl, for instance, that correspond to CVllu in the other Dravidian languages (e.g. Tamil̲ [ʋil], Malayāḷaṁ [ʋillɨ], Kannaḍa [bɪllu], Telugu [ʋillu] 'bow'), from which SoInEn deduces a Tamil̲ sound change l > ∅ / [sonorant] _ [sonorant]. Also, since the data is not loanword-annotated, SoInEn captures some loanword-specific phonetic transformations, for instance Tamil̲ s > t͡ʃ / # _ V, which can be observed in e.g. [t͡ʃuːrijən] from Sanskrit *sūrya* 'sun' (cf. Monier-Williams 1899) or [t͡ʃeptəmbəɾ] from English *September*.

Overall, the explanations SoInEn gives for the high belief non-GS sound laws it inferred are quite comprehensible, and some of them are actually justified. Many of the false positives are owed to the still rather generous `isMergeable` condition of the Gapped Frequency Trie's `merge`ing algorithm, which should be reworked for future versions.

6 Conclusion and Outlook

In this thesis, I have presented SoInEn, a PSL model for sound law inference. I aimed to reproduce the reasoning steps applied in the comparative method in first-order logic rules in order to fully automate it. In the end, not all parts of the comparative method could be performed inside PSL, and the inference had to be split into three distinct phases. This shows the high complexity of the task and indicates that computational methods will not reach the accuracy of human linguists' work very soon.

Still, given that SoInEn could not operate under the best circumstances (see the following section), the results are quite acceptable. It correctly detected most of the Dravidian proto-phoneme inventory and found several important sound laws for both Dravidian and Samoyedic. As expected, the Samoyedic vowel changes posed a challenge for proto-phoneme reconstruction, leading SoInEn to only work reasonably well on the Proto-Samoyedic consonants. Unfortunately, it missed almost all of the more specific or infrequent sound changes in both language families, and had trouble detecting the correct contexts for conditioned sound laws.

6.1 Future Work

These results highlight the need for several more or less immediate improvements to SoInEn. The Dravidian data in particular contains many loanwords, none of which are annotated as such. While older loanwords have often participated in many of the regular sound changes, the more recent ones, especially those with foreign phonotactics, can easily set even a robust sound law inference model on the wrong track when appearing in larger numbers. The input for automated sound law inference should therefore ideally be loanword annotated.

Also regarding the input data, it turns out that rich phonological transcription as in NorthEuraLex can be both a blessing (for getting a precise idea of how a word is pronounced) and a curse (for obtaining reasonable sound laws). Many of both the inferred and missing sound laws referred to sound changes that historical linguists would normally

not consider as such, but rather regard as language-specific phonological processes operating on the allophonic level. Intervocal spirantization in Tamil̲ (e.g. d̪ > ð) is such a case. More generally, the even voicing of intervocal plosives in Proto-Dravidian (as well as modern Tamil̲ and Malayāḷaṁ) is not considered phonemic in the literature. You would usually not want a sound law inference system to generate such sound laws, which would, however, require phonemically instead of phonologically transcribed input data. During development of EtInEn, we briefly considered creating a component that derives a phonemic representation for all of our word forms. However, it quickly turned out that there are indeed positional allophones that we would want to distinguish under circumstances that are not trivial to model objectively. Consider, for instance, the insertion of the nasals [ɲ] and [ŋ] before word-initial front and back vowels in Nganasan and Nenets: Strictly speaking, these are not phonemic; they do not contrast with raw word-initial vowels and their occurrence is entirely predictable. Still, the nasal insertion is cited as an essential sound law for these two languages throughout the Uralist literature (Janurik 1982; Sammallahti 1988; Mikola 2004). Objectively deriving a consistent phonemic transcription for a whole lexical database is thus a task that cannot be solved easily.

A future task that is more immediately under the control of SoInEn is the improvement of the alignments. We have seen that, even though enriched by phoneme similarity judgments, the multiple sequence alignments generated during preprocessing are sometimes quite messy. The implementation especially has problems aligning gaps, often preferring to insert or delete vowels rather than consonants, instead aligning spurious consonant sounds with vowels. This is because gaps are currently treated as phonemes as well. It might be beneficial to introduce sophisticated gap penalties instead (cf. Phillips, Janies, and Wheeler 2000).

In addition, the merging condition of the Gapped Frequency Trie needs to be refined, since it sometimes assigns high frequency to sound correspondences that are only attested once, but seemingly compatible with another very frequent partial sound correspondence. To prevent this from happening too often, I have already implemented the restriction that at least one sound change must be reflected in both merging candidates. This condition seems to work well by itself, since virtually all of the problematic cases are instances where a rare correspondence was merged with one that a linguist would consider to consist of all-equal sounds, that were however transcribed with slightly different IPA symbols (see section 5.3.2.3). The current merging condition therefore needs to be extended (rather than replaced) to distinguish such notational differences from real phonemic distinctions.

Another bottleneck for the quality of SoInEn's results was context detection. While in many cases the data was just too sparse to infer the right conditions for sound changes, the evidence would sometimes have justified the correct context, but was not properly generalized. This was usually a problem with vowels: When a sound change occurs once before, say, five different vowels, each of the individual `Pnei` observations is too infrequent to justify even generating a `Pcxt` atom. Together, however, they provide compelling evidence for a human linguist to infer a general `vowel` context. This could perhaps be resolved by "lifting" the context of the `Pnei` observations already to sound classes, composing their frequency prior out of the frequencies of the subclass observations. One must then take care to not overgeneralize the sound classes (the five vowels provide equal evidence for

`vowel`, `sonorant` and `anything`), e.g. by penalizing more general sound classes. This would be an exact reproduction of the current PSL rules for `Pcxt` atoms though, outsourcing one additional inference step originally thought for PSL.

Further improvements to context detection could be made by considering more types of contexts, e.g. referring to syllable structure. Many Samoyedic sound laws applied in coda position, which had to be modeled as two conditioned sound changes in SoInEn, one applying word-finally and the other before a consonant, the latter of which was only possible due to Samoyedic's lack of onset consonant clusters. Not a single one of these sound laws was found. Having contexts such as "onset" or "coda" could be helpful, though this would additionally require syllabification of the input data. Dravidian would also have benefited from larger contexts spanning multiple symbols or being able to refer to e.g. the nucleus of the following syllable. Other useful contexts could be reference to stress patterns (which would have to be included in the input) or vowel harmony.

Finally, the biggest issue that caused most of SoInEn's false negatives is data sparsity. The $500 - 700$ mostly incomplete cognate sets found for Samoyedic and Dravidian in the NorthEuraLex data are simply not enough for inferring precise conditioned sound laws. This insight has strong implications for future data collection enterprises. As already mentioned, NorthEuraLex is one of the larger lexical databases around. Many of its competitors require only about 200 concepts per language, since this number has been deemed ideal for phylogenetic inference (Rama and Wichmann 2018), the probably most frequent use case of these databases. It now turns out though, that we might need far more data for other applications in historical linguistics which extract more fine-grained information for individual languages, such as the highly interwoven tasks of sound law inference and proto-form reconstruction, both of which belong to List's (2019) "Open Problems".

6.2 Working with PSL

As I have outlined in section 2.4, PSL seems like a perfect match for sound law detection, promising precise and efficient inference over a model specified by logical rules directly translating the human reasoning applied during the comparative method. Unfortunately, this is often not as straightforward as anticipated.

While the very initial idea of this project was to recreate the entire comparative method in a single PSL model, with all of its heuristics and interconnectivity, perhaps even linking it directly to other EtInEn components such as loanword detection and reconstruction in the future, it quickly turned out that the resulting dependency structure is too complex even for PSL. That the final model is now separated into three distinct inference phases and many steps were outsourced to dedicated Java implementations is the consequence of this.

From my experience, I can confirm that the runtime of PSL's inference algorithm does indeed scale linearly with the number of ground rules. However, the number of ground rules can easily explode exponentially depending on how many atoms a rule connects. Usually, a significant amount of engineering is necessary to write rules in such a way that

they do not overly inflate the ground rule store, and it is often not trivial to find out why a certain rule slows down inference so much more than another. While the logical framework of PSL seemed to promise more or less straightforward translation of human reasoning strategies into a probabilistic model, in reality, formulating PSL rules often requires a mathematical way of thinking.

This is not PSL's fault, but owed to the fact that first-order logic often follows different reasoning paths than human intuition. Beltagy, Erk, and Mooney (2014) already found that the Łukasiewicz way to interpret conjunctions is counter-intuitive when dealing with fuzzy problems such as textual similarity. Similarly, implications often have different semantics for humans than in mathematical logic. When I state: "If sound correspondence X frequently occurs next to sound correspondence Y and Y is reconstructed to have evolved from a proto-vowel, then X probably evolved next to vowels", I implicitly assume that "X evolved next to vowels" is not supported by this rule in case X and Y are never observed together. In terms of logic, however, $0 \rightarrow 1 \equiv 1$, so such a rule is always supportive of the consequent. In other terms, PSL cannot directly operate under the premise "no evidence is equivalent to counter-evidence". To implement this, one has to use negative priors, which can be highly dependent on the size of the input data (see section 2.2.3).

Another problem is that, when there are several competing hypotheses, PSL becomes very indecisive. Since there is no pressure on the antecedent to be true in order for the consequent to reach an arbitrarily high belief in an implication, PSL is free to lower all atoms' belief values in order to reduce distance to satisfaction and ease the pressure on the system. One often has to explicitly encourage it to assign higher belief by way of arithmetic rules of the type `Pred(+X) = 1`. Even then, it will often find it cheaper to distribute the belief mass evenly over all competing atoms than to make a clear decision for one of them, and there is no direct way to tell it otherwise.

Despite these difficulties in designing the model, PSL has the immense advantage over e.g. "black box" machine learning approaches that it is able to comprehensibly inform about its reasoning. The Fact Viewer and its verbalization components for ground rules that was created for EtInEn makes it possible to directly inspect the model's decisions, providing interesting information not only for debugging purposes. Using the LINQS implementation's rule weight learning algorithms, it is possible to train a PSL model and explore the importance it attributes to the individual rules, which can give insights into which parts of the comparative method contribute most to its unending success.

Bibliography

Aikhenvald, Alexandra Y. and R. M. W. Dixon (2001). "Introduction." In: *Areal Diffusion and Genetic Inheritance. Problems in Comparative Linguistics*. Ed. by Alexandra Y. Aikhenvald and R. M. W. Dixon. Oxford: Oxford University Press, pp. 1–26.

Anderson, Cormac, Tiago Tresoldi, Thiago C. Chacon, Anne-Maria Fehn, Mary Walworth, Robert Forkel, and Johann-Mattis List (2018). "A Cross-Linguistic Database of Phonetic Transcription Systems." In: *Yearbook of the Poznań Linguistic Meeting*. Vol. 4. 1, pp. 21–53.

Andronov, Michail S. (2003). *A comparative grammar of the Dravidian languages*. Ed. by Dieter B. Kapp. Beiträge zur Kenntnis südasiatischer Sprachen und Literaturen 7. Wiesbaden: Harrassowitz Verlag.

Asher, Ronald E. and T. C. Kumari (1997). *Malayalam*. Ed. by Bernard Comrie. Descriptive Grammars. London/New York: Routledge.

Bach, Stephen H., Matthias Broecheler, Bert Huang, and Lise Getoor (2017). "Hinge-Loss Markov Random Fields and Probabilistic Soft Logic." In: *Journal of Machine Learning Research* 18.109, pp. 1–67. URL: http://jmlr.org/papers/v18/15-631.html.

Beltagy, Islam (2016). "Natural Language Semantics Using Probabilistic Logic." PhD thesis. University of Texas at Austin.

Beltagy, Islam, Katrin Erk, and Raymond Mooney (2014). "Probabilistic soft logic for semantic textual similarity." In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, Maryland: Association for Computational Linguistics, pp. 1210–1219.

Bouchard-Côté, Alexandre, David Hall, Thomas L. Griffiths, and Dan Klein (2013). "Automated reconstruction of ancient languages using probabilistic models of sound change." In: *Proceedings of the National Academy of Sciences* 110.11, pp. 4224–4229.

Burrow, Thomas and Murray Barnson Emeneau (1984). *A Dravidian Etymological Dictionary*. 2nd ed. Oxford: Clarendon Press.

Campbell, Lyle (2013). *Historical Linguistics. An introduction*. 3rd ed. Edinburgh: Edinburgh University Press.

Crowley, Terry and Claire Bowern (2010). *An Introduction to Historical Linguistics*. 4th ed. New York: Oxford University Press.

Daneyko, Thora and Christian Bentz (2019). "Click languages tend to have large consonant inventories: Implications for language evolution and change." In: *Modern Human Origins and*

Dispersal. Ed. by Yonatan Sahle, Hugo Reyes-Centeno, and Christian Bentz. Vol. 2. Word, Bones, Genes, Tools: DFG Center for Advanced Studies. Tübingen: Kerns Verlag, pp. 315–329.

Dellert, Johannes (2017). "Information-Theoretic Causal Inference of Lexical Flow." PhD thesis. University of Tübingen.

Dellert, Johannes (2018). "Combining Information-Weighted Sequence Alignment and Sound Correspondence Models for Improved Cognate Detection." In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, pp. 3123–3133.

Dellert, Johannes, Thora Daneyko, Alla Münch, Alina Ladygina, Armin Buch, Natalie Clarius, Ilja Grigorjew, Mohamed Balabel, Hizniye Isabella Boga, Zalina Baysarova, Roland Mühlenbernd, Johannes Wahle, and Gerhard Jäger (2020). "NorthEuraLex: a wide-coverage lexical database of Northern Eurasia." In: *Language Resources and Evaluation* 54, pp. 273–301.

Dellert, Johannes and Gerhard Jäger, eds. (2017). *NorthEuraLex*. Version 0.9. URL: `http://www.northeuralex.org/`.

Deng, Lingjia and Janyce Wiebe (2015). "Joint Prediction for Entity/Event-Level Sentiment Analysis using Probabilistic Soft Logic Models." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, pp. 179–189.

Dunn, Michael, ed. (2015). *Indo-European Lexical Cognacy Database*. URL: `http://ielex.mpi.nl/` (visited on 06/13/2020).

Greenhill, Simon J., Robert Blust, and Russell D. Gray (2008). "The Austronesian basic vocabulary database: from bioinformatics to lexomics." In: *Evolutionary Bioinformatics* 4, pp. 271–283.

Hajdú, Péter (1988). "Die samojedischen Sprachen." In: *The Uralic Languages. Description, History and Foreign Influences*. Ed. by Denis Sinor. Leiden: E. J. Brill, pp. 3–40.

Hale, Mark (2007). "The Regularity of Sound Change." In: *Historical Linguistics. Theory and Method*. Blackwell Textbooks in Linguistics. Oxford: Blackwell Publishing, pp. 124–146.

Hall, David and Dan Klein (2010). "Finding Cognate Groups using Phylogenies." In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1030–1039.

Hämäläinen, Mika and Jack Rueter (2019). "Finding Sami Cognates with a Character-Based NMT Approach." In: *Proceedings of the 3rd Workshop on Computational Methods for Endangered Languages*. Vol. 1, pp. 39–45.

Hammarström, Harald, Robert Forkel, and Martin Haspelmath, eds. (2019). *Glottolog 4.1*. URL: `http://glottolog.org` (visited on 02/27/2020).

Harrison, S. P. (2003). "On the Limits of the Comparative Method." In: *The Handbook of Historical Linguistics*. Ed. by Brian D. Joseph and Richard D. Janda. Blackwell Handbooks in Linguistics. Oxford: Blackwell Publishing, pp. 213–243.

Haspelmath, Martin and Uri Tadmor, eds. (2009). *WOLD*. Leipzig: Max Planck Institute for Evolutionary Anthropology. URL: `https://wold.clld.org/`.

Helimski, Eugene (1993). "Prasamodijskie ə̑ i ə: praural'skie istočniki i nganasanskie refleksy." In: *Hajdú Péter 70 éves*. Ed. by Marianne Bakró-Nagy and Enikő Szíj, pp. 125–133.

Helimski, Eugene (2005). "The 13th Proto-Samoyedic vowel." In: *Mikola-konferencia 2004*. Ed. by Beáta Wagner-Nagy, pp. 27–39.

Hruschka, Daniel J, Simon Branford, Eric D Smith, Jon Wilkins, Andrew Meade, Mark Pagel, and Tanmoy Bhattacharya (2015). "Detecting regular sound changes in linguistics as events of concerted evolution." In: *Current Biology* 25.1, pp. 1–9.

Irikov, S. I. (1988). *Slovar' sel'kupsko-russkij i russko-sel'kupskij*. Leningrad: Prosveščenie.

Jäger, Gerhard, Johann-Mattis List, and Pavel Sofroniev (2017). "Using support vector machines and state-of-the-art algorithms for phonetic alignment to identify cognates in multi-lingual wordlists." In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Vol. 1. Valencia, pp. 1205–1216.

Janhunen, Juha (1977). *Samojedischer Wortschatz. Gemeinsamojedische Etymologien*. Castrenianumin toimitteita 17. Helsinki: Suomalais-Ugrilainen Seura, Helsingin yliopisto.

Janhunen, Juha (1998). "Samoyedic." In: *The Uralic Languages*. Ed. by Daniel Abondolo. London/New York: Routledge, pp. 457–479.

Janurik, Tamás (1982). "Szamojéd hangmegfelelések." In: *Nyelvtudományi Közlemények*. Ed. by Péter Hájdu and Károly Rédei, pp. 41–89.

Kaiping, Gereon and Marian Klamer, eds. (2019). *LexiRumah 3.0.0 –- A lexical database of Lesser Sunda languages*. URL: `http://lexirumah.model-ling.eu/` (visited on 06/14/2020).

Keane, Elinor (2004). "Tamil." In: *Journal of the International Phonetic Association: Illustrations of the IPA* 34.1, pp. 111–116.

Kiparsky, Paul (2003). "The Phonological Basis of Sound Change." In: *The Handbook of Historical Linguistics*. Ed. by Brian D. Joseph and Richard D. Janda. Blackwell Handbooks in Linguistics. Oxford: Blackwell Publishing, pp. 313–342.

Kittel, Ferdinand (1894). *A Kannaḍa-English Dictionary*. Mangalore: Basel Mission Book & Tract Depository.

Köllner, Marisa and Johannes Dellert (2016). "Ancestral State Reconstruction and Loanword Detection." In: *Proceedings of the Leiden Workshop on Capturing Phylogenetic Algorithms for Linguistics*. Ed. by Christian Bentz, Gerhard Jäger, and Igor Yanovich.

Krishnamurti, Bhadriraju (2003). *The Dravidian Languages*. Cambridge Language Surveys. Cambridge: Cambridge University Press.

Kroonen, Guus (2013). *Etymological Dictionary of Proto-Germanic*. Ed. by Alexander Lubotsky. Vol. 2. Leiden Indo-European Etymological Dictionary Series. Leiden/Boston: Brill.

Labat, Sofie and Els Lefever (2019). "A Classification-Based Approach to Cognate Detection Combining Orthographic and Semantic Similarity Information." In: *Recent Advances in Natural Language Processing 2019*, pp. 603–611.

Lewis, M. Paul, Gary F. Simons, and Charles D. Fennig, eds. (2015). *Ethnologue: Languages of the World*. URL: `https://www.ethnologue.com/18` (visited on 02/27/2020).

LINQS Research Group (2018). *Probabilistic Soft Logic (PSL)*. Version 2.1.0. URL: `https://psl.linqs.org/`.

List, Johann-Mattis (2012). "LexStat: Automatic detection of cognates in multilingual wordlists." In: *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*. Association for Computational Linguistics. Avignon, pp. 117–125.

List, Johann-Mattis (2019a). "Automatic inference of sound correspondence patterns across multiple languages." In: *Computational Linguistics* 45.1, pp. 137–161.

List, Johann-Mattis (2019b). *Open problems in Computational Historical Linguistics*. Invited talk presented at the 24th International Conference of Historical Linguistics. Canberra.

List, Johann-Mattis, Cormac Anderson, Tiago Tresoldi, Simon J. Greenhill, Christoph Rzymski, and Robert Forkel (2019). *Cross-Linguistic Transcription Systems*. Version 1.2.0. Max Planck Institute for the Science of Human History, Jena. URL: `https://clts.clld.org/` (visited on 09/13/2019).

List, Johann-Mattis, Philippe Lopez, and Eric Bapteste (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*. Berlin, pp. 599–605.

List, Johann-Mattis and Steven Moran (2013). "An Open Source Toolkit for Quantitative Historical Linguistics." In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, pp. 13–18.

Liu, Shulin, Kang Liu, Shizhu He, and Jun Zhao (2016). "A Probabilistic Soft Logic Based Approach to Exploiting Latent and Global Information in Event Classification." In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI-16. Phoenix, Arizona: AAAI Press, pp. 2993–2999.

Lowe, John B. and Martine Mazaudon (1994). "The Reconstruction Engine: A Computer Implementation of the Comparative Method." In: *Computational Linguistics* 20.3, pp. 381–417.

Mennecier, Philippe, John Nerbonne, Evelyne Heyer, and Franz Manni (2016). "A Central Asian language survey: Collecting data, measuring relatedness and detecting loans." In: *Language Dynamics and Change* 6.1, pp. 57–98.

Menon, A. Sreedhara (1978). *Cultural Heritage of Kerala. An Introduction*. Cochin: East-West Publications.

Mi, Chenggang, Yating Yang, Lei Wang, Xi Zhou, and Tonghai Jiang (2018a). "A Neural Network Based Model for Loanword Identification in Uyghur." In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. Miyazaki, pp. 3575–3579.

Mi, Chenggang, Yating Yang, Lei Wang, Xi Zhou, and Tonghai Jiang (2018b). "Toward better loanword identification in uyghur using cross-lingual word embeddings." In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, pp. 3027–3037.

Mi, Chenggang, Yating Yang, Xi Zhou, Lei Wang, Xiao Li, and Tonghai Jiang (2016). "Recurrent Neural Network Based Loanwords Identification in Uyghur." In: *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation*. Seoul, pp. 209–217.

Mikola, Tibor (1988). "Geschichte der samojedischen Sprachen." In: *The Uralic Languages. Description, History and Foreign Influences*. Ed. by Denis Sinor. Leiden: E. J. Brill, pp. 219–263.

Mikola, Tibor (2004). *Studien zur Geschichte der samojedischen Sprachen*. Ed. by Beáta Wagner-Nagy. Szeged: SzTE Finnisch-Ugrisches Institut.

Minett, James W. and William S.-Y. Wang (2003). "On detecting borrowing." In: *Diachronica* 20.2, pp. 289–330.

Monier-Williams, M. (1899). *A Sanskrit-English Dictionary. Etymologically and philologically arranged with special reference to Cognate indo-european languages*. URL: https://www.sanskrit-lexicon.uni-koeln.de/scans/MWScan/2020/web/index.php.

Normanskaja, Julia (2018). "Reconstruction of the 14th and 15th Proto-samoyedic Vowels." In: *NordSci Conference on Social Sciences. Conference Proceedings*. Vol. 1. 1. Helsinki, pp. 295–305.

Notredame, Cédric, Desmond G. Higgins, and Jaap Heringa (2000). "T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment." In: *Journal of Molecular Biology* 302.1, pp. 205–217.

Oakes, Michael P. (2000). "Computer Estimation of Vocabulary in a Protolanguage from Word Lists in Four Daughter Languages." In: *Journal of Quantitative Linguistics* 7.3, pp. 233–243.

Phillips, Aloysius, Daniel Janies, and Ward Wheeler (2000). "Multiple Sequence Alignment in Phylogenetic Analysis." In: *Molecular Phylogenetics and Evolution* 16.3, pp. 317–330.

Prakash, Ashok, Arpit Sharma, Arindam Mitra, and Chitta Baral (2019). "Combining Knowledge Hunting and Neural Language Models to Solve the Winograd Schema Challenge." In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, pp. 6110–6119.

Rama, Taraka (2016). "Siamese convolutional networks for cognate identification." In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, pp. 1018–1027.

Rama, Taraka and Johann-Mattis List (2019). "An automated framework for fast cognate detection and Bayesian phylogenetic inference in computational historical linguistics." In: *57th Annual Meeting of the Association for Computational Linguistics*. Florence.

Rama, Taraka, Johannes Wahle, Pavel Sofroniev, and Gerhard Jäger (2017). "Fast and unsupervised methods for multilingual cognate clustering." In: *arXiv:1702.04938*.

Rama, Taraka and Søren Wichmann (2018). "Towards identifying the optimal datasize for lexically-based Bayesian inference of linguistic phylogenies." In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, pp. 1578–1590.

Rankin, Robert L. (2003). "The Comparative Method." In: *The Handbook of Historical Linguistics*. Ed. by Brian D. Joseph and Richard D. Janda. Blackwell Handbooks in Linguistics. Oxford: Blackwell Publishing, pp. 183–212.

Rospocher, Marco (2018). "An Ontology-driven Probabilistic Soft Logic Approach to Improve NLP Entity Annotations." In: *Proceedings of the 17th International Semantic Web Conference*. Monterey, California, pp. 144–161.

Sammallahti, Pekka (1988). "Historical Phonology of the Uralic Languages. With Special Reference to Samoyed, Ugric, and Permic." In: *The Uralic Languages. Description, History and Foreign Influences*. Ed. by Denis Sinor. Leiden: E. J. Brill, pp. 478–554.

Schiffman, Harold F. (1999). *A Reference Grammar of Spoken Tamil*. Cambridge: Cambridge University Press.

Schuchardt, Hugo (1885). *Ueber die Lautgesetze. Gegen die Junggrammatiker*. Berlin: Verlag von Robert Oppenheim.

Setälä, Eemil (1901). "Über transskription der finnisch-ugrischen sprachen. Historik und vorschläge." In: *Zeitschrift für Finnisch-Ugrische Sprach- und Volkskunde* 1. Ed. by Eemil Setälä and Kaarle Krohn, pp. 15–52.

Sridhar, Dhanya, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker (2015). "Joint models of disagreement and stance in online debate." In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 116–125.

Staal, J. F. (1963). "Sanskrit and Sanskritization." In: *The Journal of Asian Studies* 22.3, pp. 261–275.

Tuisk, Tuuli (2010). "Some Aspects of Quantity in Central Veps." In: *Linguistica Uralica* 46.4, pp. 241–249.

Turchin, Peter, Ilia Peiros, and Murray Gell-Mann (2010). "Analyzing genetic connections between languages by matching consonant classes." In: *Journal of Language Relationship* 3, pp. 117–126.

Wang, Wei-Chung and Lun-Wei Ku (2016). "Identifying Chinese lexical inference using probabilistic soft logic." In: *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. San Francisco, California, pp. 737–743.

Wang, Wei-Chung and Lun-Wei Ku (2017). "Enabling Transitivity for Lexical Inference on Chinese Verbs Using Probabilistic Soft Logic." In: *Proceedings of The 8th International Joint Conference on Natural Language Processing*. Taipei, pp. 110–119.

Wichmann, Søren, Eric W. Holman, and Cecil H. Brown, eds. (2020). *The ASJP Database*. Version 19. URL: `https://asjp.clld.org/`.

A Source Code

The source code of SoInEn, as it was presented in this thesis, is available from:

- `https://github.com/tdaneyko/etinen-soundlaws-standalone/releases/tag/ma-thesis` (Standalone SoInEn without the EtInEn GUI)
- `https://github.com/jdellert/etinen-full/releases/tag/ma-thesis-tdaneyko` (EtInEn (GUI) with integrated SoInEn)

As of now, EtInEn is still under development, and the repositories are private. They will be made public once the first version of EtInEn is released. If you wish to gain pre-release access to the code, please contact

- Thora Daneyko (thora.daneyko@student.uni-tuebingen.de) for standalone SoInEn, or
- Johannes Dellert (johannes.dellert@uni-tuebingen.de) for EtInEn or the EtInEn-independent PSL wrapper with the Fact Viewer.

Sound Transition Matrix and Gold Standard

The files containing the sound transition matrix described in section 4.2.6 is located (in both repositories) under `etinen-soundlaws/src/main/resources/sound-transition-matrix.tsv`.

The gold standard files for Dravidian and Samoyedic can be found under `etinen-soundlaws/src/test/resources/eval/`. The suffix `-nelex` marks the NorthEuraLex-specific gold standard files that were used during the evaluation discussed in chapter 5. The `-phonemic` files are directly derived from the sound laws presented in chapter 3, whereas in the `-phonemic-single` files, phoneme clusters were dissolved to fit the `Plaw` format, but no NorthEuraLex-specific transformations were performed yet.

B Sound Classes

The following table lists all sound classes used by SoInEn. How these classes were obtained is described in section 4.2.5. In addition, each of the single sounds is a sound class as well, containing only itself.

Lvl	Class	Members
1	anything	vowel, consonant
1	anything	obstruent, sonorant
2	obstruent	stop, fricative, affricate
2	sonorant	nasal, approximant, trill, tap, vowel
3	vowel	unrounded, rounded
3	vowel	open, near-open, open-mid, mid, close-mid, near-close, close
3	vowel	front, near-front, central, near-back, back
3	consonant	voiced, voiceless
3	consonant	stop, fricative, affricate, nasal, approximant, trill, tap
3	consonant	labial, dental, alveolar, post-alveolar, alveolo-palatal, palatal, retroflex, velar, uvular, glottal, epiglottal, pharyngeal
3	consonant	continuant, occlusive
3	consonant	sibilant
3	consonant	liquid
4	labial	bilabial, labio-dental, labio-palatal, labio-velar
4	continuant	fricative, approximant, trill
4	occlusive	stop, affricate, nasal
4	liquid	rhotic, lateral
5	affricate	c͡ç, d͡z, d͡ʑ, d͡ʒ, p͡f, q͡χ, t͡s, t͡ɕ, t͡ɬ, t͡ʃ, ɖ͡ʐ, ʄ͡ʝ, ʈ͡ʂ
5	alveolar	d, d͡z, l, n, r, s, t, t͡s, t͡ɬ, z, ɬ, ɮ, ɹ, ɻ, ɾ
5	alveolo-palatal	d͡ʑ, t͡ɕ, ɕ, ʑ
5	approximant	j, l, l̪, w, ɥ, ɭ, ɹ, ɻ, ʊ, ʍ, ʏ

Lvl	Class	Members
5	back	o, u, ɑ, ɒ, ɔ, ɤ, ɯ, ʌ
5	bilabial	b, m, p, ɸ, β
5	central	ɐ, ɘ, ə, ɜ, ɨ, ɵ, ʉ
5	close	i, u, y, ɨ, ɯ, ʉ
5	close-mid	e, o, ø, ɘ, ɤ, ɵ
5	dental	d̪, l̪, n̪, s̪, t̪, ð, θ
5	epiglottal	ʜ, ʡ
5	fricative	f, h, s, s̪, v, x, z, ç, ð, ħ, ɕ, ɣ, ɦ, ɬ, ɮ, ɸ, ʁ, ʂ, ʃ, ʐ, ʑ, ʒ, ʕ, ʜ, j, β, θ, χ
5	front	a, e, i, y, æ, ø, œ, ɛ
5	glottal	h, ɦ, ʔ
5	labio-dental	f, p͡f, v, ɱ, ʋ
5	labio-palatal	ɥ
5	labio-velar	w, ʍ
5	lateral	l, l̪, t͡ɬ, ɬ, ɭ, ɮ, ɪ, ʎ
5	mid	ə
5	nasal	m, n, n̪, ŋ, ɱ, ɲ, ɳ, ɴ
5	near-back	ʊ
5	near-close	ɪ, ʊ, ʏ
5	near-front	ɪ, ʏ
5	near-open	æ, ɐ
5	open	a, ɑ, ɒ
5	open-mid	œ, ɔ, ɛ, ɜ, ʌ
5	palatal	c, c͡ç, j, ç, ɟ, ɟ͡ʝ, ɲ, ʎ, ʝ
5	pharyngeal	ħ, ʕ
5	post-alveolar	d͡ʒ, t͡ʃ, ʃ, ʒ
5	retroflex	ɖ, ɖ͡ʐ, ɭ, ɳ, ɻ, ɽ, ʂ, ʈ, ʈ͡ʂ, ʐ
5	rhotic	r, ɹ, ɪ, ɻ, ɽ, ɾ, ʀ, ʁ
5	rounded	o, u, y, ø, œ, ɒ, ɔ, ɵ, ʉ, ʊ, ʏ
5	sibilant	d͡z, d͡ʑ, d͡ʒ, s, s̪, t͡s, t͡ɕ, t͡ʃ, z, ɕ, ɖ͡ʐ, ʂ, ʃ, ʈ͡ʂ, ʐ, ʑ, ʒ
5	stop	b, c, d, d̪, k, p, q, t, t̪, ɖ, ɟ, g, ɢ, ʈ, ʔ, ʡ
5	tap	ɪ, ɽ, ɾ
5	trill	r, ʀ
5	unrounded	a, e, i, æ, ɐ, ɑ, ɘ, ə, ɛ, ɜ, ɤ, ɨ, ɪ, ɯ, ʌ
5	uvular	q, q͡χ, ɢ, ɴ, ʀ, ʁ, χ
5	velar	k, x, ŋ, g, ɣ
5	voiced	b, d, d̪, d͡z, d͡ʑ, d͡ʒ, j, l, l̪, m, n, n̪, r, v, w, z, ð, ŋ, ɖ, ɖ͡ʐ, ɟ, ɟ͡ʝ, g, ɢ, ɣ, ɥ, ɦ, ɭ, ɮ, ɱ, ɲ, ɳ, ɴ, ɹ, ɪ, ɻ, ɽ, ɾ, ʀ, ʁ, ʋ, ʎ, ʐ, ʑ, ʒ, ʕ, j, β

Lvl	Class	Members
5	voiceless	c, c͡ç, f, h, k, p, p͡f, q, q͡χ, s, s̪, t, t̪, t͡s, t͡ɕ, t͡ɬ, t͡ʃ, x, ç, ħ, ɕ, ɬ, ɸ, ʂ, ʃ, ʈ, ʈ͡ʂ, ʍ, ʔ, н, ʢ, θ, χ

C Gold Standard Files Used in Evaluation

C.1 Dravidian

```
#inventory
i e a ə o u p t̪ t ʈ tʃ k b d̪ d ɖ dʒ g m n̪ ɳ ɲ l ɭ ɾ ʊ ɻ j

#soundlaws
//lang   proto target   left  right set   comment

kan   i  ɛ   0  *a    with /a/ in next syllable
kan   u  ɔ   0  *a    with /a/ in next syllable
tel   i  e  0  *a    with /a/ in next syllable
tel   u  o  0  *a    with /a/ in next syllable
tam,mal  e  i  0  *a    with /a/ in next syllable
tam,mal  o  u  0  *a    with /a/ in next syllable
kan   e  ɪ   0  *high    with high vowel in next syllable
kan   o  ʊ   0  *high    with high vowel in next syllable

kan   p  h  #  0
kan   m  -  0  b
kan   b  ʊ  m  0
tel   b  -  m  0
kan   p  -  p  0  kan-pp
kan   p  -  0  p  kan-pp
tel   p  -  p  0  tel-pp
tel   p  -  0  p  tel-pp
tam,mal  m  -  0  p

tam   d̪     vowel 0
mal   d̪  n̪  n̪  0
kan   t̪  -  t̪  0  kan-tt
kan   t̪  -  0  t̪  kan-tt
tel   t̪  -  t̪  0  tel-tt
tel   t̪  -  0  t̪  tel-tt
```

```
tam,mal  ɳ  -  0  ʈ

tam,mal,kan d  ɾ  vowel 0
tel   d  ɽ   vowel 0
tam   n  ɳ  0  0
tam   d  ɳ  n  0
mal   n  ɳ  0  0
mal   d  ɳ  n  0
kan   n  ɳ  0  0
kan   d  ɖ  n  0
tel   n  ɳ  0  0
tel   d  ɖ  n  0
kan   t  ɾ  t  0  kan-_t_t-ɾ
kan   t  ɾ  0  t  kan-_t_t-ɾ
kan   t  ʈ  t  0  kan-_t_t-t
kan   t  ʈ  0  t  kan-_t_t-t
tel   t  ʈ  t  0  tel-_t_t
tel   t  ʈ  0  t  tel-_t_t
kan   t  -  t  0  kan-_t_t
kan   t  -  0  t  kan-_t_t
tel   t  -  t  0  tel-_t_t
tel   t  -  0  t  tel-_t_t
tam,mal  n  -  0  t

mal,tel  tʃ   tɕ   0  0
kan   tʃ   s  #  0
tam,kan  dʒ   s  vowel 0
tel   dʒ   ʂ  vowel 0
mal   dʒ   ɲ  ɲ  0
kan   tʃ   -  t  0  kan-cc
kan   t  -  0  tʃ   kan-cc
tel   tʃ   -  t  0  tel-cc
tel   t  -  0  tʃ   tel-cc
tam,mal  ɲ  -  0  tʃ

kan   ʈ  -  ʈ  0  kan-.t.t
kan   ʈ  -  0  ʈ  kan-.t.t
tel   ʈ  -  ʈ  0  tel-.t.t
tel   ʈ  -  0  ʈ  tel-.t.t
tam,mal  ɳ  -  0  ʈ

tam   k  tʃ   #  front
mal,tel  k  tɕ   #  front
kan,tel  k  g  #  *son     irregularly, when following syllable starts with
    sonorant consonant
tam   g  ɣ   vowel 0
mal   g  ŋ  ŋ  0
kan   k  -  k  0  kan-kk
kan   k  -  0  k  kan-kk
tel   k  -  k  0  tel-kk
tel   k  -  0  k  tel-kk
```

```
tam,mal  ŋ  -  0  k

kan,tel  ɲ  n̪  0  0

tel   ɳ  n̪  0  0

kan   ʊ  b  #  0

tel   ɭ  l  0  0

kan,tel  ɾ  r  0  0

kan   ɻ  r  0  consonant
kan   ɻ  l  0  0
tel   ɻ  r  consonant   0
tel   ɻ  ɖ  0  0


// Transcription-specific sound laws:

// Short /a/ is transcribed [ə] in non-initial syllables in Tamil and
    Malayalam, [ə] except word-finally in Kannada, and ʌ[] everywhere in
    Telugu and can hence be reconstructed as [ə] instead of [a]
tam   ə  a  #  0  tam-a
tam   ə  a  *#C   0  tam-a
mal   ə  a  #  0  mal-a
mal   ə  a  *#C   0  mal-a
kan   ə  a  0  #
tel   ə  ʌ   0  0

// Tamil non-initial /u/ is transcribed ʉ[]
tam   u  ʉ   0  #  tam-u
tam   u  ʉ   *non-init   0  tam-u
// Tamil and Kannada insert [w] before word-initial [u], [o] and [j] before
    word-initial [i], [e]
tam,kan  -  w  #  back
tam,kan  -  j  #  front

// Malayalam word-final *u is [ɨ]
mal   u  ɨ  0  #
// Malayalam /kk/ is audibly palatalized after /i/ and in some other instances
mal   -  ʲ  k  0
// Also, Malayalam /ɾ/ is generally palatalized
mal   -  ʲ  ɾ  0
// Malayalam [n̪] is alveolar [n] intervocally and word-finally
mal   n̪  n  vowel vowel
mal   n̪  n  0  #

// Kannada short /i/, /e/, /o/ and /u/ are transcribed lax except word-finally
kan   i  ɪ   0  consonant   kan-i
kan   i  ɪ   0  vowel kan-i
```

```
kan   e  ɛ   0  consonant   kan-e
kan   e  ɛ   0  vowel kan-e
kan   o  ɔ   0  consonant   kan-o
kan   o  ɔ   0  vowel kan-o
kan   u  ʊ   0  consonant   kan-u
kan   u  ʊ   0  vowel kan-u
```

C.2 Samoyedic

```
#groups
[i,e] i e

#inventory
i y ɯ u e ø ə ɤ o æ ɒ p t k ʧ ʂ m n ɲ ŋ s l r w j

#soundlaws
//lang   proto target   left  right set   comment

nio   æ  ɐ  0  0
enf   æ  e  0  0
yrk   æ  ɑ  0  0
sel   æ  ø  0  0

nio   ɒ  o  #  0
nio   ɒ  u  0  0
enf,yrk  ɒ  ɑ  0  0
sel   ɒ  a  0  0

nio   e  ɐ  0  0     irregular

nio   ø  u  0  0
enf,yrk  ø  o  0  0
sel   ø  y  0  0

nio   i  ɨ  labial   0

nio   y  i  0  0
enf,yrk  y  u  0  0

nio   ɤ  ɨ  0  r
nio   ɤ  ɐ  0  0
enf   ɤ  u  labial   0
enf   ɤ  i  0  0
yrk   ɤ  e  0  0
sel   ɤ  ɨ  0  0

nio   o  u  0  0

enf   ɯ  u  labial   0
nio,enf,yrk,sel   ɯ  i  0  0
```

```
nio   ə  e  palatal  0
enf   ə  o  0  0
yrk,sel  ə  ɑ  0  0

nio   -  ʲ  alveolar front
yrk   -  ʲ  consonant   front

nio   p  h  0  vowel
nio   p  b  0  consonant
nio   p  b  0  #
enf,yrk  p  b  vowel vowel
enf   p  -  0  consonant
enf   p  -  0  #

nio   t  t  0  vowel nio-t
nio   t     vowel vowel nio-t
enf,yrk  t  d  vowel vowel
enf   t  -  0  consonant
enf   t  -  0  #
nio,yrk  t  ʔ  0  consonant
nio,yrk  t  ʔ  0  #

enf   s  -  0  consonant
enf   s  -  0  #
nio,yrk  s  ʔ  0  consonant
nio,yrk  s  ʔ  0  #

nio   ʈ͡ʂ   s  0  [i,e] nio-c-pal
nio   ʈ͡ʂ   s  0  ʲ  nio-c-pal
nio   ʈ͡ʂ   t  0  vowel nio-c
nio   ʈ͡ʂ      vowel vowel nio-c
enf,yrk  ʈ͡ʂ   d  vowel vowel
enf   ʈ͡ʂ   -  0  consonant
enf   ʈ͡ʂ   -  0  #
sel   ʈ͡ʂ   ɕ  0  [i,e]
nio,yrk  ʈ͡ʂ   ʔ  0  consonant
nio,yrk  ʈ͡ʂ   ʔ  0  #
nio,enf,yrk,sel   ʈ͡ʂ   t  0  0

nio   k  s  0  [i,e] nio-k-pal
nio   k  s  0  ʲ  nio-k-pal
enf   k  s  0  front
enf   k  h  vowel back
enf   k  -  0  consonant
enf   k  -  0  #
yrk   k  s  0  front yrk-k-pal
yrk   k  s  0  ʲ  yrk-k-pal
yrk   k  x  0  back
sel   k  ɕ  0  [i,e]
sel   k  q  0  open
```

```
sel   k  q  0  close-mid
nio,yrk  k  ʔ  0  consonant
nio,yrk  k  ʔ  0  #

nio   m  -  0  #     multi-syllable words only
enf   m  -  vowel vowel
enf   m  -  0  consonant
enf   m  -  0  #
yrk   m  w  vowel vowel

nio   n  -  0  #     multi-syllable words only
nio   n  ŋ  0  #     single-syllable words only
enf   n  -  0  consonant
enf   n  -  0  #
yrk   n  ʔ  0  consonant
yrk   n  ʔ  0  #

nio,enf  ɲ  -  vowel vowel
yrk   ɲ  j  vowel vowel

nio   ŋ  -  0  #     multi-syllable words only
enf   ŋ  -  vowel 0
yrk   ŋ  ʔ  0  consonant
yrk   ŋ  ʔ  0  #

nio   -  ɲ  #  [i,e]
nio   -  ŋ  #  vowel
yrk   -  ɲ  #  front
yrk   -  ŋ  #  back

enf   l  r  vowel vowel
enf   l  -  0  consonant
enf   l  -  0  #

enf   r  -  0  consonant
enf   r  -  0  #

nio   w  b  0  0
enf   w  b  #  0
enf,sel  w  -  vowel vowel
yrk   w  j  0  front
yrk   w  b  vowel vowel
sel   w  k  #  0

nio   j  d  0  vowel nio-j-d
nio   j  d  0  ʲ  nio-j-d
enf   j  d  #  0
yrk   j  -  0  consonant
yrk   j  -  0  #
sel   j  t͡ɕ   0  vowel
```