Phonotactic Structures in Swedish

A Data-Driven Approach

Felix Hultin

Department of Linguistics

Magister thesis 15 credits
Computational Linguistics
Spring 2017
Tutor: Mats Wirén
Examinator: Bernhard Wälchli
Reviewer: Robert Östling

Stockholms universitet

Phonotactic Structures in Swedish

A Data-Driven Approach

Abstract

Ever since Bengt Sigurd laid out the first comprehensive description of Swedish phonotactics in 1965, it has been the main point of reference within the field. This thesis attempts a new approach, by presenting a computational and statistical model of Swedish phonotactics, which can be built by any corpus of IPA phonetic script. The model is a weighted trie, represented as a finite state automaton, where states are phonemes linked by transitions in valid phoneme sequences, which adds the benefits of being probabilistic and expressible by regular languages. It was implemented using the Nordisk Språkteknologi (NST) pronunciation lexicon and was used to test against a couple of rulesets defined in Sigurd relating to initial two consonant clusters of phonemes and phoneme classes. The results largely agree with Sigurd's rules and illustrated the benefits of the model, in that it effectively can be used to pattern match against phonotactic information using regular expression-like syntax.

Keywords

Phonotactics, computational phonology, trie, finite automata, pattern matching, regular languages

Sammanfattning

Ända sedan Bengt Sigurd lade fram den första övergripande beskrivningen av svensk fonotax 1965, så har den varit den främsta referenspunkten inom fältet. Detta examensarbete försöker sig på en ny infallsvinkel genom att presentera en beräkningsbar och statistisk modell av svensk fonotax som kan byggas med en korpus av fonetisk skrift i IPA. Modellen är en viktad trie, representerad som en ändlig automat, vilket har fördelarna av att vara probabilistisk och kunna beskrivas av reguljära språk. Den implementerades med hjälp av uttalslexikonet från Nordisk Språkteknologi (NST) och användes för att testa ett par regelgrupper av initiala två-konsonant kluster av fonem och fonemklasser definierad av Sigurd. Resultaten stämmer till större del överens med Sigurds regler och visar på fördelarna hos modellen, i att den effektivt kan användas för att matcha mönster av fonotaktisk information med hjälp av en liknande syntax för reguljära uttryck.

Nyckelord

Fonotax, beräkningsbar fonologi, trie, ändlig automat, mönstermatchning, reguljära språk

Contents

1. Introduction

Ever since Swedish phonotactics was first laid out by Bengt Sigurd in his doctoral thesis *Phonotactic Structures of Swedish* (Sigurd, 1965), the research area has been largely confined to the results of his more than 50-year-old endeavor. Indeed, in the latest accounts of Swedish Phonotactics, such as in Tomas Riad's book *The Phonology of Swedish* (Riad, 2013, ch. 12), Sigurd's work is still referred to as the main point of reference.

Meanwhile, in the area of computational phonology, the mathematical model of finite state automata has become essential for representing phonological observations, recently coupled with statistical models to predict phonological information. On a different note, a vast digital lexicon of word entries by Språkteknologi Holding, including pronunciation data, was released in 2011 to the public by the Norwegian Språkbanken, giving access to a phonological resource previously not available.

In the light of and inspired by these separate developments, this thesis will present a computational, data-driven, statistical, model of Swedish phonotactics, which can be built by any corpus of phonetic script, based on the International Phonetic Alphabet (IPA) (International Phonetic Association, 1999). The model is a weighted trie, represented as a probabilistic finite automaton, where states are phonemes linked by transitions in valid phoneme sequences, representing the likelihood of one phoneme following another. I will investigate the models' computational benefits, especially in the context of phonotactic research, and, as a proof-of-concept, test some sample rules defined in Sigurd's thesis against corresponding generated results by the model.

With this research, I hope to lay out and demonstrate the need for this type of computational model, which I will argue is an important infrastructure for data-driven research of Swedish phonotactics.

2. Background

This thesis is based on, on the one hand, the linguistic research area of phonology, especially phonotactics in Swedish, and, on the other hand, the computational and mathematical theories, which will be used to compute a phonotactic model. Therefore, I will, in this section, cover both of these areas, in order to put the need for a computational, statistical model into perspective and to lay out the necessary theory for implementing it.

2.1. Phonology

Phonology is the study of how sounds are organized in natural languages. This stands in contrast to phonetics, which studies the physiological, aerodynamic and acoustic characteristics of speech-sounds (Catford, 1988). Although both disciplines are in many ways dependent on each other, it can generally be said that phonetics studies continuous aspects of sound, which phonology then organizes into discrete systems of natural languages. This continuous and discrete relation between the two disciplines is important, as it will reappear as we get into the International Phonetic Alphabet.

Phonology itself consists of two fundamental parts: The classification of phonemes and the study of the arrangement and combination of defined phonemes, referred to as phonotactics (Sigurd, 1965). Although this thesis focuses on the latter, one cannot discard the importance of the classification part, seeing that it is a prerequisite for studying and understanding phonotactic structures in the first place. Therefore, I will lay out a brief overview of the phonemes in Swedish and put them in the context of the International Phonetic Alphabet (IPA), the primary phonetic notation used in this thesis, in order to properly transition into Swedish phonotactics.

2.1.1. Phonemes in Swedish

In central Swedish, the variety treated in this thesis, there are 35 different phonemes, of which 17 are vowels and 18 are consonants. An important feature of Swedish phonology is that both vowels and consonants (except for /ɕ/ and /h/) can be either short or long. For vowels, this is illustrated in table 1.

Phoneme	Long Vowel	Short vowel	Orthography	Long example	Short example
/i/	[iː]	[ɪ]	<i>	*bit* [biːt] 'piece'	*vinn* [vɪnː] 'win'
/y/	[yː]	[ʏ]	<y>	*byt* [byːt] 'change'	*fynd* [fʏnːd] 'find'
/e/	[eː]	[ɛ̝]	<e>	*bet* [beːt] 'bit'	*sett* [sɛ̝tː] 'seen'
/ɛ/	[ɛː]	[ɛ̝]	<ä>	*mät* [mɛːt] 'measure'	*sätta* [sɛ̝tːa] 'to set'
/ø/	[øː]	[ø]	<ö>	*böta* [bøːta] 'to pay a fine'	*lönn* [lønː] 'maple tree'
/ʉ/	[ʉ̞ː]	[ɵ]	<u>	*muta* ˈmʉ̞ːta 'to bribe'	*lund* [lɵnːd] 'grove'
/u/	[uː]	[ʊ]	<o>	*bot* [buːt] 'cure'	*bonde* [bʊnːdɛ] 'farmer'
/o/	[oː]	[ɔ]	<å>	*båt* [boːt] 'boat'	*fond* fɔnːd 'fund'
/ɑ/	[ɑː]	[a]	<a>	*mat* [mɑːt] 'food'	*vann* [vanː] 'won'

Table 1: The vowels of central Swedish, listed with their phonemes, their long and short variants, the corresponding orthography most often used in written text, an example word of a long vowel, and an example word of a short vowel.

Here we see that every vowel has a long and short phoneme pair. This is because replacing one with the other changes the meaning of the utterance. For example, the only phonological difference between the word *mat* (food) and *matt* (weak, listless) is that of the phoneme [a] and [ɑː], but still changes the semantic meaning of the word. For consonants, however, this is not the case. Although every consonant (except for /ɕ/ and /h/)), also have long and short versions, as illustrated in table 2, they are not treated as separate phonemes in Swedish. This is because, unlike the example above, switching between a short

and a long version of the consonant does not change a word's meaning, even if one version might sound unfamiliar to a native speaker (Riad, 2013, ch. 3). In other words, long and short varieties are phonemic for vowels, while allophonic for consonants.

		labial, labiodental	dental, alveolar	alveolar, palatal	velar	glottal
oral stop	s.g.	p pᵘ	t tᵘ		k kᵘ	
	voice	b bᵘ	d dᵘ		g gᵘ	
fricative	s.g.	f fᵘ	s sᵘ	ɕ		h
fric./retroflex			s ʂ			
fric./approx.	voice	v vᵘ		ʝʝᵘ		
nasal stop		m mᵘ	n nᵘ		ŋ ŋᵘ	
lateral			l lᵘ			
apical trill			r rᵘ			

Table 2: The consonants of Swedish, tabulated by manner of articulation and, in some cases, the absence of a voicedness. Both long and short versions are given, the latter with a raised more, e.g. gᵘ.

2.1.2. International Phonetic Alphabet

The International Phonetic Alphabet (IPA) is a standardized format for representing spoken languages. Having been around since 1888, it is the most widely used phonetic alphabet within the linguistic community and is included in the Unicode standard (Unicode Consortium, 1997). There are mainly two types of symbols in IPA: **Letters**, which represent distinctive sounds in speech, and **suprasegmentals**, which represent sound features stretching across many distinctive sounds. Additionally, letters can be modified with **diacritics** to add certain features.

Although IPA has the word *phonetic* in it, it is largely a phonological endeavor, in that it organizes speech sounds in a discrete domain, namely an alphabet. This becomes clear when looking at the previous tables (table 1 and 2) of Swedish phonemes, which illustrate the usage of IPA symbols in representing sounds of speech. For vowels, we have the letters *i, y, e, ɛ, ø, ʉ, o, ɑ, ɪ, ʏ, ɛ, ø, ɵ, ʊ, ɔ, a*, the diacritics ˧, ˨, and the suprasegmental *ː* and for consonants the letters *p, t, k, b, d, g, f, s, v,ɕ, h, v, j, m, n, ŋ, l, r* and the diacritics ˛ and μ. What we see here is that even though a letter can often be used to represent one specific phoneme in a language, there is not a one-to-one correspondence between letters in IPA and phonemes of a specific language. Actually, many IPA letters can be used to signify the same phoneme of a language or vice versa, often by combining them with diacritics or suprasegmentals. An example of this can be seen with the letter ɛ, which is used to represent two different phonemes, namely [ɛ̝] and [ɛː]. This is due to the fact that IPA is a multilingual alphabet, meant to represent sounds of all natural languages, which means that many different phones will signify the same phoneme in one language, while being different phonemes in another language (International Phonetic Association, 1999).

2.1.3. Distinctive Features

Besides assigning speech sounds with specific symbols, as IPA does, they can also be grouped together based on certain phonological criteria, what in phonology is called **distinctive features**. These features include every distinguishable category there is to find in speech, but must be binary classified, in the sense that a feature can only be either true or false. Again, these are discrete classifications of what in phonetics might be continuous.

An example of distinct features can be seen in previous table 2 for consonants, where column values are distinctive features of a certain kind, namely manner and place of articulation. This is a quite common tabularization when wanting to emphasize the relation between pairs of distinctive features. Another common way to represent distinctive features is in a feature based table, where the rows con-

tain IPA letters (often phonemes in a language), and the columns distinctive features. Table 3, is an example of this kind of table:

phoneme	consonant	vowel	labial	stop	fricative	voiced	voiceless
b	+	-	+	+	-	+	-
p	+	-	+	+	-	-	+
f	+	-	+	-	+	-	+

Table 3: Distinctive features table of the phonemes b, p and f, indicating the presence of the features consonant, vowel, labial, stop, fricative, voiced, voiceless.

Here we see the three phonemes /b/, /p/, /f/ as rows and the distinctive features *consonant, vowel, labial, stop, fricative, voiced and voiceless* as columns. A value, which can either be $+$ or $-$, thus indicates the presence or absence of a distinctive feature of a certain letter (or phoneme) (Catford, 1988, p. 189).

Now distinctive features themselves can be organized into even more groups, i.e. major class features, laryngeal features, manner features, place features and vowel space, creating a kind of taxonomy of distinctive features. This, however, will not be treated in this thesis.

2.1.4. Phonotactics in Swedish

Although Swedish linguistics had had a long linguistic tradition throughout the 20th century, a comprehensive Swedish phonotactics was only laid out in 1965 with the publication of Bengt Sigurd's doctoral thesis *Phonotactic Structures in Swedish* (Haugen, 1967; Sigurd, 1965). Inspired by the structural school of linguistics, Sigurd was able to take the classification of Swedish phonemes, which at that time did exist, and account for their arrangement and combination by many different means of analysis, such as position analysis, combination analysis and sequence analysis. Since then, Sigurd's thesis, together with the review and commentary of Bengt Loman (Riad, 2013; Loman, 1967), has served as a point of reference for Swedish phonotactics.

In broad terms, Sigurd lays out a fundamental description of Swedish phonotactics by first defining permissible sequences of phonemes, i.e. the arrangement and combination of phonemes, and then identifying general phonological patterns which account for them. The analysis of permissible sequences are divided up based on the position of the sequences. These are 1) initial sequences, phoneme sequences that appear at the beginning of the word, 2) final sequences, phoneme sequences that appear at the end of the word, and 3) medial sequences, phoneme sequences that appear in between initial and final sequences. Mostly initial and final sequences are covered and even though vowels are briefly accounted for in some chapters (Sigurd, 1965, ch. VI, VII, VII), the thesis focuses mostly on permissible consonant sequences. For example, initial and final sequences are mostly analyzed based on initial consonant clusters.

Sigurd compiles his models using an empirical, data-driven approach. He uses the 9th edition of *Svenska Akademiens ordlista* (SAOL) (Akademien, 1950), containing about 200 000 orthographic entries, to identify permissible sequences. Presumably, although this is not explicitly mentioned in the thesis, the words are transcribed by the author himself, since the entries in SAOL 9 are exclusively orthographic and not phonetically transcribed. Foreign words are not excluded from the data, but if there is a sequence, which only appear in foreign words, they are treated as exceptions and are not included in the final models. Completely unassimilated words, such as *manager* and *outsider*, are however not included at all. What is important to note, is that Sigurd does not include exact frequencies of the sequences or any statistical information about the structures he identifies and limits the amount of examples to no more than three (Sigurd, 1965, section 1.5). This seems quite understandable, given the amount of extra work it would take to keep track of all frequencies and the scarce availability of computer resources at the time.

2.1.5. Initial Sequences in Phonotactic Structures in Swedish

Given the comprehensiveness of Sigurd's work, covering all the results of his endeavors would be beyond the scope of this thesis. I will, therefore, limit myself to the results relevant to this thesis, namely those of initial sequences.

As mentioned above, Sigurd analyzes sequences by many different means. The basis for these different analyses can be found in the sequence diagram, see figure 1, which lays out all permissible sequences of consonants in the three first positions.

In the light of the information given by the sequence diagram, many patterns are observed by Sigurd. In relation to *sequences of specific phonemes*, these rules are formulated (Sigurd, 1965, section 2.21, p. 41):

a) h, ɕ, ɧ, j, l, r can [only] be followed by a vowel

b) m, n can be followed by a vowel or by *j*

c) d, t can be followed by a vowel or by *v* or *r*

d) b, p can be followed by a vowel or by *r, l, j*

Furthermore, in relation to *permissible sequences of phoneme classes* (or distinctive features), these rules for non-permissible sequences were also formulated (Sigurd, 1965, section 2.23, p. 53):

stops-stops	(*ptkbdg-ptkbdg*)
nasals-nasals	(*mn - mn*)
fricatives-fricatives	(*sfvj-sfvj*, exceptions: *sv-fj*)
liquids-liquids	(*lr-lr*)
labials-labials	(*pbfmv-pbfmv*)
dentals-dentals	(*stdnlr-stdnlr*, exceptions: *st, sn, sl, tr, dr*)
palatals-palatals	(*kgj-kgj*)
nasals-liquids	(*mn-lr*)
stops-nasals	(*ptkbdg-mn*, exceptions: *kn, gn*)

Here we see, for example, that a labial-labial combination is not allowed in Swedish, unlike in German for example, where a /pf/ combination is allowed as in the words *Pfarrer* (pastor) or *Pferd* (horse). These phoneme classes are clearly not the only distinctive features for consonants, but they do provide informative insights to the combination possibilities of Swedish phonemes.

2.1.6. Remarks

Even though *Phonotactic Structures of Swedish* indeed is an impressive and ground-breaking work, in that it laid out an extensive description of phonotactics in Swedish, through what seems to have been the result of an intense manual effort, it has some important methodological limitations. One is the lack of transparency of the process which generated the sequences. Even though the study is an empirical one, Sigurd does not lay out the method of which he uses to analyze the SAOL dictionary. Only three quoted examples are at most presented for each sequence, leaving a small set of evidence for the claims made. Furthermore, with an absence of transition frequencies, it makes it hard to know to which extent certain sequences are outliers and which are an essential part of the phonotactic description. Of course, foreign words are noted and excluded if necessary, but even with Swedish words one might want to know how common these sequences actually are and by that fact justify whether they belong in the phonotactical

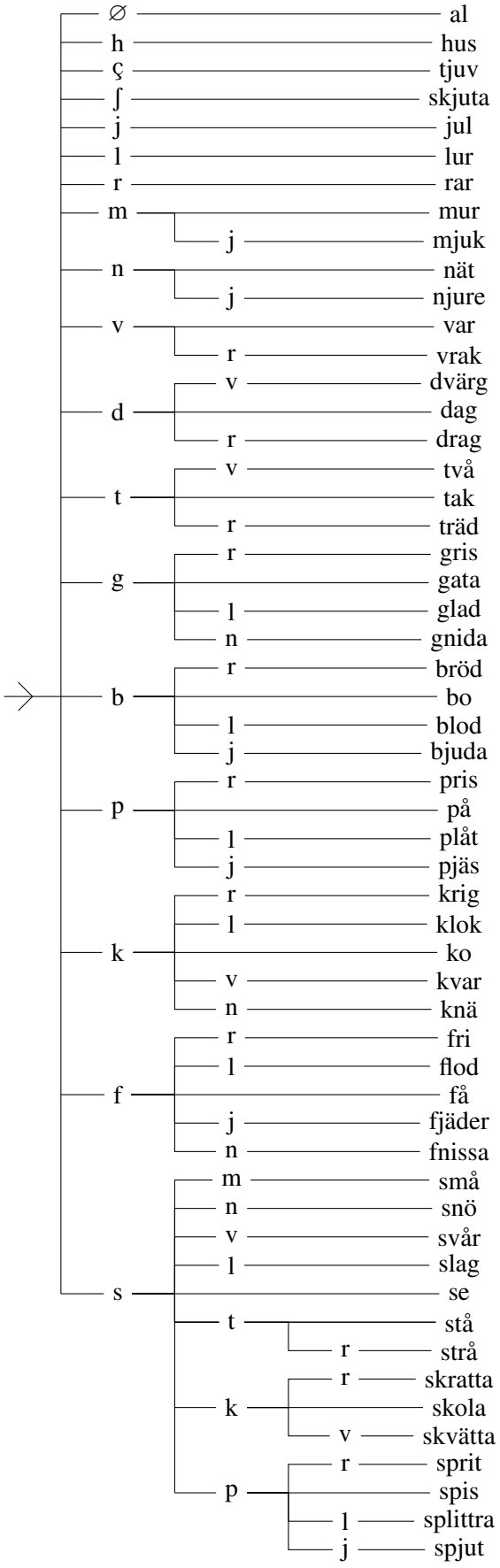


Figure 1: Word initial sequences diagram, as it appears in Sigurd (1965, p. 42), illustrating permissible consonant sequences of phonemes at the beginning of a word. Vowel phonemes are assumed to always be permissible at any position.

description or not. Finally, because these data, including the transcriptions of SAOL, are not included, it makes it very difficult to replicate Sigurd's results, of which there has hardly been any attempt.

Naturally, many of these limitations can be explained by the lack of technology at the time of publication. However, in the light of this, coupled with the fact that Sigurd's endeavor is a product of a more than 67-year-old 9th edition of SAOL, there is clearly a need for a fully transparent and traceable model of phonotactics, one that can easily be adapted to any set of data.

2.2. Computational Theory

In this section, I will lay out a brief overview of the mathematical and computational models relevant for this thesis. The order is important, because they reflect the steps that will be taken in the implementation laid out in section 4.

2.2.1. Finite Automata

A **finite automaton** is a model in mathematics and computer science, which allows for the modeling of so called **abstract machines**. These machines consist of **states** and ways in which to move between them, in order to accept or reject inputs of sequences of symbols, also referred to as **transitions**. The idea of finite machines was first introduced by McCulloch and Pitts (1943) and has been proven useful to represent many different types of observations, especially sequences of characters and strings. They have, thus, become an important concept within computational linguistics, where analyzing text is of fundamental importance. A mathematical representation of a finite automaton, summarized by Jurafsky and Martin (2000, p. 62), is given below:

$Q = q_0 q_1 q_2 \dots q_{N-1}$ a finite set of N states

Σ a finite **input alphabet** of symbols

q_0 the **start state**

F the set of **final states**, $F \subseteq Q$

$\delta(q,i)$ the **transition function** or transition matrix between states. Given a state $q \in Q$ and an input symbol $i \in \Sigma$, $\delta(q,i)$ returns a new state $q' \in Q$. δ is thus a relation from $Q \times \Sigma$ to Q;

By defining these properties above, one can define an automaton, or abstract machine, to recognize or reject specific types of inputs. For example, to define an automaton, which recognizes all the valid sub-sentences of the sentence *I don't play much sports*, one could define an input alphabet of English words as Σ = {*I, don't, play, much sports*}, the states as $Q = q_0, q_1, q_2, q_3, q_4, q_5$, the final states as $F = q_3, q_4, q_5$ and a start state q_0, representing the beginning of the sentence. Finally, with a transition matrix, illustrated in table 4, the permissible transitions between the states can be defined:

The matrix in table 4 shows the possible ways to move inside the automaton. The rows contain the states and the columns contain the possible inputs in the automaton. The value of a state and input symbol pair, thus, shows the possible transition(s) from the state given the input symbol. The symbol $\emptyset$ means there is no possible transition given the input.

To illustrate a finite automaton and its permissible transitions more clearly, a so called state diagram can be used. Figure 2 shows such a diagram for the above defined automaton to recognize sub-sentences.

Here, states are represented as nodes in a graph and the transitions as links between them. A circle inside a node indicates it being a final state. By reading it, we see that within the finite automaton only the words 'I', 'don't', 'play', 'much', 'sports' can be input and that the the only acceptable sequence of symbols are the sentences *I don't play much sports*, *I play much sports*, *I don't play much*, *I play much* and *I don't play*. Meanwhile, sentences such as *I much sports* or *sports I play* are rejected by

	Input				
State	*I*	*don't*	*play*	*much*	*sports*
q_0	q_1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	q_2	q_3	$\emptyset$	$\emptyset$
q_2	$\emptyset$	$\emptyset$	q_3	$\emptyset$	$\emptyset$
q_3	$\emptyset$	$\emptyset$	$\emptyset$	q_4	$\emptyset$
q_4	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	q_5
q_5	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Table 4: The transmission matrix of a finite automaton to recognize valid sub-sentences of the string *I don't play much sports*.

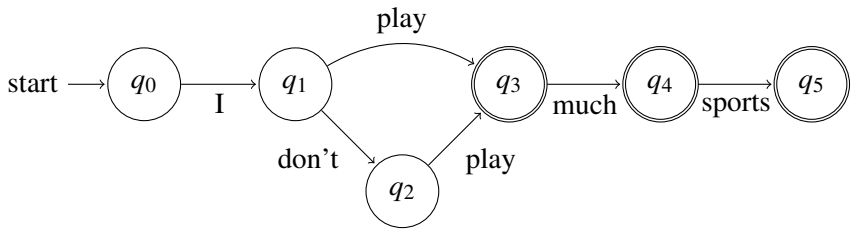


Figure 2: A state diagram illustrating a finite automaton to recognize valid sub-sentences of the words sequence *I [don't] play much sports*. Nodes represents the states ($q_0, q_1, q_2, q_3, q_4, q_5$) and arrows the permissible transitions.

the automaton because of the impossibility to move within the automaton as to recognize their input. Furthermore, a sentence like *I **do** play **soccer*** would not even be possible, because it contains symbols that are not defined in the alphabet. Naturally, one could allow for the latter inputs by extending their symbols to the alphabet, but until then it remains a mathematical contradiction.

2.2.2. Regular Languages and Pattern Matching

Finite automatons can be expressed by what are called **regular languages**, a subset of **formal languages**, used extensively within computer science as well as computational linguistics. Actually, mathematically, they are equivalent, meaning both can be used to express one another, as proven by Stephen Cole Kleene (Kleene, 1951).

A standardized way to define regular languages, is with what are called **regular expressions**. They provide a syntax for defining sequences of characters, expressing a finite automaton, and have been proven to be effective for the purposes of **pattern matching**. This is especially true with pattern matching of texts, where regular expressions are written to match certain types of strings in a text or corpus.

The syntax contains a keyboards set of characters, usually those encoded according to a standard like Unicode or ASCII, and a number of so called **metacharacters** used to describe further generic string patterns. Some of the most important of these metacharacters are . (dot), which matches any character, +, which matches preceding elements one or more times, and [], which matches any pattern character defined inside the brackets.

To illustrate the usage of regular expressions, we can take the regular expression `[a-zA-Z]+(ing|ed)`. This regular expression is intended to match words ending with the suffixes *-ing* or *-ed*. It does this by firstly using square brackets [ ] to match any lower case *a-z* or upper case *A-Z* character in the alphabet and the operator + to match any repetition of such pattern. Secondly, the parentheses ( ) group the next expression, meant to be either the suffix *-ing* or *-ed*, where the choice between the suffixes is expressed with the | 'or' pipeline (Jurafsky & Martin, 2000, ch. 2).

By formulating this regular expression, it can be used to match words in a text. Figure 3 illustrates words matched by the regular expression in a text snippet from an online article[1]: *rooted, suspecting, Beijing, using* and *allowed*. As the results indicate, many useful matches are returned, but a regular expression might also not always match the expressions the user intended. In this case, even though many words with real derivational suffixes of *-ing* or *-ed* were matched, a false positive match word *Beijing* was also matched. This goes to show that a regular expression might not always be perfect, but is a powerful tool for a user to filter intended matches.

> However, there are also deep-rooted doubts, with some suspecting Beijing is using its "win-win" project as a ploy to lure less powerful nations into its economic orbit and boost its geopolitical power. Privately, western diplomats voice concerns about China's true intentions and how much involvement non-Chinese companies will be allowed to have in Belt and Road projects. Only one G7 leader, the Italian prime minister Paolo Gentiloni, is in Beijing for Xi's summit.

Figure 3: Words matched by the the regular expression, highlighted with orange, in snippet from an English article.

Now, as mentioned before, since regular languages are equivalent to finite automata, the above expression can naturally also be expressed as a finite automaton. The state diagram in figure 4 does exactly that.

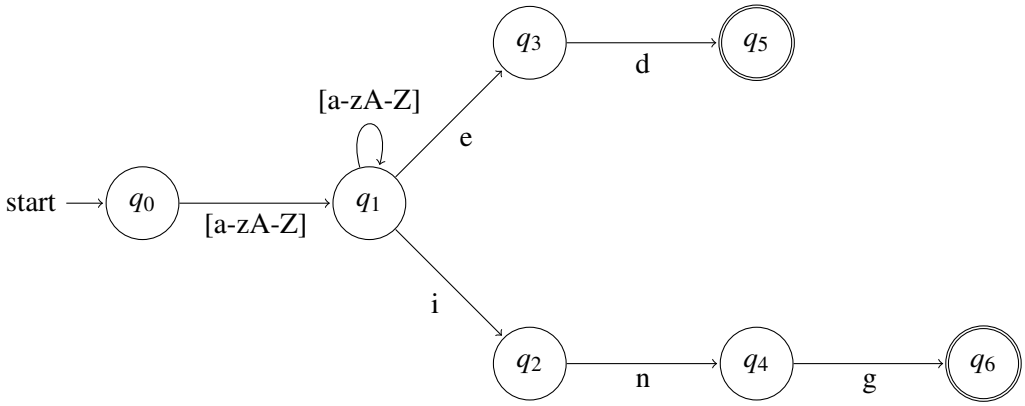


Figure 4: A state diagram, illustrating the regular expression `[a-zA-Z]+(ing|ed)` as a finite automaton.

This diagram illustrates the possible transitions the characters of the word must take for it to be matched by the regular expression. As we can see, in order to reach the end states of either q_5 or q_6, the initial part of the word would first have to be matched by at least one character, which is not any of the two suffixes, to then be matched by one of the suffixes *-ing* or *-ed*.

2.2.3. Trie

Introduced by De La Briandais (1959) a **trie** is a tree data structure, often used to store strings of characters or words in a dictionary. In a trie nodes make up the characters of words, wherein a word can be generated by following a path from the root node to a leaf. What is crucial about a trie is that words with the same prefix share the same prefix character nodes. For this reason, a trie is also called a **prefix tree** (Brass, 2008, p. 336). Figure 5, illustrates a trie for words starting with the prefix *phon-*:

[1]Tom Philips (2017, May 14) China's Xi lays out $900bn Silk Road vision amid claims of empire-building. *The Guardian*. Retrieved from https://www.theguardian.com

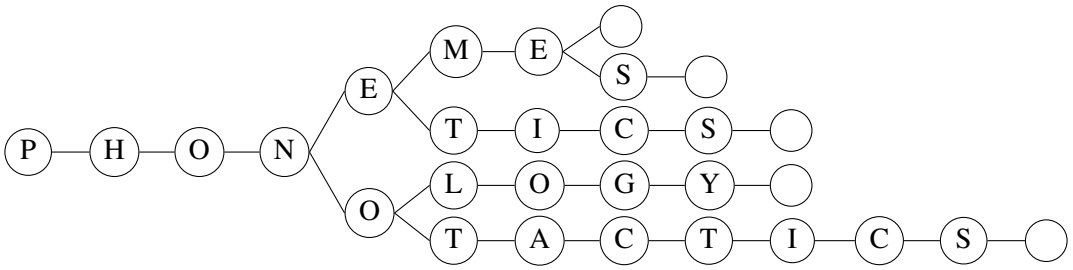


Figure 5: Example of a letter trie, storing words of the prefix *phon-*.

Here we see that the words *phoneme, phonetics, phonology* and *phonotactics* are stored in the trie. Furthermore, we can observe that words with the same prefix share the prefix characters memory. For example, all words share the same memory for their prefix *phon-*, the words *phoneme* and *phonemes* share the *e* character in the prefix *phone-*, and the words *phonetics*, *phonotactics* and *phonology*, share the same character *o* in the *phono-* prefix.

Tries are, unsurprisingly, most known for representing dictionaries. Because prefixes of words only occur once in the data structure, they allow for quick lookup speed even when dealing with a large amount of entries. This has been shown to useful for autocomplete and spell checking features found in many different applications (Brass, 2008, ch. 8).

2.2.4. Computational Phonology

The idea that phonological observations can be modeled with regular languages is not new to this thesis. Inspired by the Sound Pattern of English (SPE) model of phonology laid out by Chomsky and Halle, C.D Johnson as well as Kaplan and Kay showed independently as early as in the 70's the connection between SPE rules and finite state transducers (Johnson, 1970; Kaplan & Kay, 1994). Furthermore, in the related field of computational morphology, finite state automata models have been used to model rules of morphemes and their morphological structure. One of the most significant contributions to this has been in the area of finite state morphology, in which whole systems and programming languages for the purposes of formalizing morphological structures have been made (Beesley & Karttunen, 2003).

Supplementing the developments of finite-state phonology, another strand of development emerged during the late 80's and 90's, where statistical models were applied for various different purposes, such as neural network for recognizing syllables (Gasser, 1992), minimum description length to automatically identify syllable boundaries in phonemically transcribed texts (Ellison, 1994), and Markov models for speech recognition to map speech recording to phonetic transcriptions (Garofolo, Lamel, Fisher, Fiscus, & Pallett, 1993; Bird, 2002).

Meanwhile, the trie data structure has most noticeably been utilized within the area of automatic learning, in particular for unsupervised morphological induction of suffix paradigms (Schone & Jurafsky, 2000).

This thesis intends to follow and combine these developments, by applying the same computational theory but for the purposes of phonotactic description. This is something which, as far as the author knows, has been left unexplored in the field of computational phonology, let alone in the field of Swedish phonotactics.

3. Aims and Research Questions

The aim of this study is to create a computational model of Swedish phonotactics for the purposes of studying Swedish phonotactics. In the light of this, three research questions have been formulated:

1) How can a computational model of Swedish Phonotactics be represented to enable efficient extraction of phonotactic information?

2) What benefits can a computational model of Swedish phonotactics add to modern Swedish phonotactics?

3) Where do computationally generated phoneme sequences differ from initial two consonant clusters laid out by Sigurd in terms of phonemes and phoneme classes (distinctive features)?

The two first questions are of more general nature, which answers are meant to put the model into the context of the relevant research areas, while the third question, in the light of Sigurd's two consonant cluster rules laid out in 2.1.5, adds to the previous ones by elucidating the usage of the model in real phonotactic research.

4. Method and Data

In this section, a representation of phonotactic observations in terms of a trie and the way in which it is populated and normalized for the purposes of this study will be laid out. Furthermore, methods used to visualize the trie in this thesis will be covered. Finally, a method for extracting phonotactic and statistical information for the model to answer the third research question will be defined.

4.1. Data

The data used to compute the statistical model for this thesis is the lexicon for Swedish by Nordisk Språkteknologi (NST), a Norwegian language technology company, which went bankrupt in 2003 (Nasjonalbiblioteket, 2011, p. 1). The data was released to the public in 2011 by the Norwegian Språbanken and is, as far as the author know, the only publicly available digital pronunciation dictionary of Swedish and has not been used in the academic literature, which has been an important motivation for conducting this thesis.

The NST lexicon is an extensive table with as many as 51 columns and 927 167 words. Of these words, about 250 000 are manually annotated, while 677 000 are machine generated inflections of the original words. Of the 250 000 original words, 86 000 come from a frequency based lexicon by the NST company, 100 000 from Telia related projects and the rest from other projects in the industry. The words are, like in Sigurd's (1965) work, based on the central Swedish variety, particularly the Stockholm variety. They are also of a general character and are not domain-specific (Nasjonalbiblioteket, 2011, p. 6-7).

The columns in the table cover a vast array of information, most notably phonetic transcription, orthographic form, lemmas and morphological features, all of which have a varying degree of coverage (see the documentation for a more detailed enumeration). For all intents and purposes, this thesis will only be concerned with the transcription column and to a limited extent the orthographic form column, both of which are included in all entries. Nonetheless, other columns could be utilized in future projects, in order to add more information to the statistical model or to modify it in some significant ways (see section 6) (Nasjonalbiblioteket, 2011, p. 6).

The transcriptions in the NST lexicon are encoded in SAMPA(Wells, 1997), a computer-readable phonetic script, which uses seven-bit encoded ASCII characters to represent IPA phonetic notation. Since all SAMPA transcriptions have their IPA equivalences, converting one to the other is quite straightforward.

4.1.1. Data Filtering

For this thesis, the lexicon has been filtered by certain criteria. Firstly, because the focus is exclusively on initial sequences (see research question 3), all non-controlled automatically generated inflected words, marked with the INFLECTED value in the inflector role column (column number 31), have been filtered out. Doing this does not only reduce the number of entries significantly but also gives a somewhat fairer probability distribution, seeing that words with a lot of inflections do not get counted more times than those with fewer inflections. Secondly, in order for the results not to be cluttered with too many foreign words, only entries marked with the value SWE in the language code orthography column (column number 6) are included. As will be shown in the results, however, this did not stop certain foreign words to be included. Thirdly, entries marked with the value ABBR or ACCR in the acronym/abbreviation column (column number 9) were not included. After filtering, the data was reduced to 83 662 entries.

4.2. A Trie Representation of Phonotactics

Relating to the introduction on finitie automa and tries in section 2.2, I will in this section introduce the formal probabilistic acyclic finite automaton representation of a weighted trie used in this study to compute a computational model of phonotactics:

$p_0 =$ a special **start state** that is not associated with observations.

$P = p_1 p_2 \dots p_N$ a set of N phoneme states, where p_{ij} represents a phoneme state at position S_j in a valid sequence of phonemes.

$A = a_{01}, a_{02} \dots a_{nn}$ a **transition probability matrix** A, each a_{ij} representing the probability of moving from phoneme state p_i to state p_j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall p_i$

$PA = \{q_x, q_y, \dots q_i\}$ a set $PA \subset P$ of legal **accepting states**, where q_i represents the last phoneme in a valid sequence of phonemes, i.e. a word $W = p_1 p_2 \dots q_i$

Here, we see that phonemes are not represented as states themselves, but rather as their particular subsequent position in relation to previous phonemes of a valid sequence of phonemes. Starting states and accepting states correspond to the beginning and ending of words, such that an initial state p_i is always the first phoneme of a word and an accepting state q_i is always the last phoneme of a word.

Now that the mathematical model has been defined, the next section will go into how the lexicon can be used to computationally implement the model.

4.2.1. Implementation of Phonotactic Model

The trie was implemented in the Python 3 (van Rossum, 1995) program PHONO-STRUCT[2] and includes four steps, listed below:

1) Convert the SAMPA encoded transcriptions entries in the lexicon into their IPA equivalences (pre-processing).

2) Extract a trie from the IPA transcriptions, such that all shared prefixes share the same memory and where the trie represents all valid phoneme sequences in the lexicon.

3) Extend the trie into a probabilistic weighted tree, such that each edge is assigned a probability between $0 \dots 1$ and where all probabilities from an edge of a parent node to its children sums up to 1.

4) Compress the trie into a probabilistic acyclic finite state automaton, such that P stores the nodes in the trie, A the weighted edges in the tree and PA the nodes which are parents to end-of-word (empty) nodes.

Having completed these steps with the lexicon, we end up with an automaton consisting of 361 251 states/vertices and 361 250 transitions/edges. So far, the trie is stored in memory and can only be accessed through internal programmatic means. To make the information more accessible to both programmers and non-programmers, I will in the next section lay out a way to extract information from the model by the means of pattern matching.

4.2.2. Extracting Information from the Phonotactic Model with Pattern Matching

Now that the trie has been represented and compressed into a probabilistic finite automaton, we can reap the benefits of it representing transition probabilities between phonemes and being recognizable by stochastic languages, including regular languages. In other words, the probability of certain phoneme sequences can be calculated, regular expressions to match certain types of phoneme sequences can be formulated or both can even be combined. This means, it is possible to extract all kinds of phonotactic information from the model, which is important, considering the amount of states and transitions it has.

[2]Source code available at GitHub: https://github.com/felixhultin/phono-struct (accessed: 2017-05-20, commit-id: 8e13d8bd7aefd9023820fc298464ad196c87e6c0).

For the purposes of this thesis, a search function SEQUENCE(PHONEMES) has been implemented. Given a defined phoneme sequence search pattern, SEQUENCE(PHONEMES) finds the phoneme sequences in the phonotactic model, which match the pattern. This sequence pattern is a regular expression, supporting a subset of the syntax found in some of the most common standards, such as POSIX (Group, 2008) and Perl (Wall, 2000). Table 5 summarizes the symbols used for the formulated search patterns in this thesis:

	Syntax	Description
Meta characters	`^`	Matches the beginning of the string.
	`[...]`	Matches a single character defined within the brackets.
	`+`	A unary operator to indicate the presence of a character class.
	`-`	A unary operator to indicate the absence of a character class.
Character classes	`consonant`	Consonant letters (p, t, k, b, d, g, f, s, v...)
	`vowel`	Vowel letters (i, y, e, ɛ, ø, ʉ, o, ɑ, ɪ, ʏ, ɛ, ø...)
	`stop`	Stop letters (p, t, k, b, d, g)
	`nasal`	Nasal letters (m, n)
	`fricative`	Fricative letters (s, f, v, j)
	`liquid`	Liquid letters (l, r)
	`labial`	Labial letters (p, b, f, m, v)
	`dental`	Dental letters (s, t, d, n, l, r)
	`palatal`	Palatal letters (k, g, j)

Table 5: Regular expression syntax used in this thesis.

As indicated in the table, the syntax symbols are divided up into metacharacters and character classes. What makes this syntax different from other regular expression standards is the extension of distinctive feature character classes. Furthermore, these character classes can be modified with a binary unary operator of either a + or - sign to match letters having these distinctive features or not. This might differ from other standards, where especially the + is only reserved for matching one or more occurence of a character, but importantly complements the phonological notation of distinctive (see section 2.1.3).

Finally, the search patterns formulated in this thesis also uses character class subtraction, a notation supported by some standards such as the XML Schema (Yergeau, 2010) and XPath (Clark, DeRose, et al., 1999). It allows for excluding characters inside a bracket notation *after* other character classes have been defined. This is done by subtracting some characters inside separate brackets with the - operator from another character class: `/[class-[subtractedclass]/.`

To illustrate the usage of this syntax, we will take the example of a rule concerning three initial phoneme consonant cluster in Swedish phonotactics. It states that a three initial consonant cluster can only start with the phoneme /s/ (Riad, 2013, p. 283). To see whether this is valid, we need to formulate a sequence search pattern, in which the first three phonemes are all consonants. The following search pattern does exactly that:

```
^[+consonant-vowel][+consonant-vowel][+consonant-vowel]
```

Here, three seperate bracket expressions `[...]` with the distinctive feature classes `+consonant` and `-vowel` are formulated to match a consonant cluster and the metacharacter ^ is used to indicate that the pattern must be at the beginning of the sequence.

By using this search pattern as input for the SEQUENCE function, all phoneme sequences with an initial three consonant cluster in the lexicon will be returned. We will continue with the results of this search pattern in the next subsection, where the question of visualization will be discussed.

4.3. Visualizing the Phonotactic Model

Having a computational phonotactic model implemented, the next important step is to visualize it. Seeing that the model has a size of about 360 000 states/vertices and was developed for descriptive purposes, its capacity to be adequately visualized is an important feature next to effective information extraction. To this purpose, the D3.js (Bostock, Ogievetsky, & Heer, 2011) JavaScript visualization library has been used to generate interactive graphical representations of the model. To give an example, in figure 6, we see the visual results of the search pattern for initial three consonant clusters laid out above:

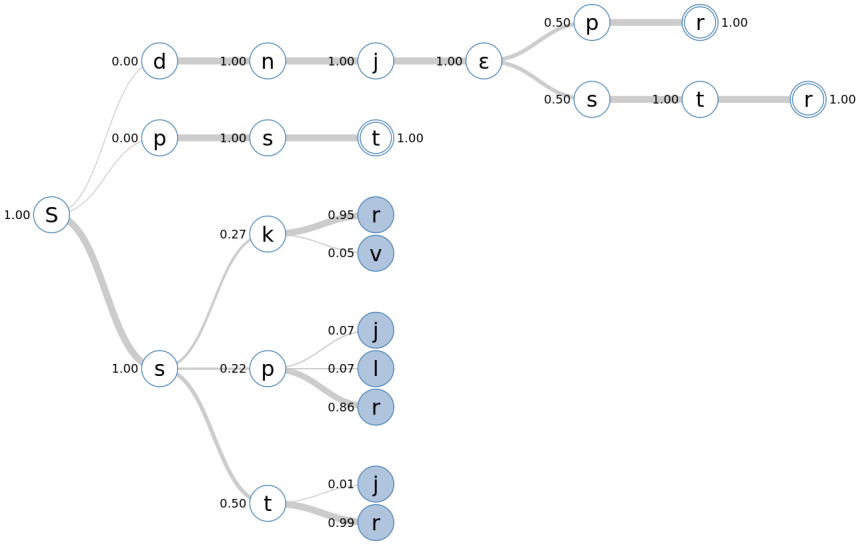


Figure 6: The resulting visualization of the resulting phonotactic model of the sequence search pattern **`/^[+consonant-vowel][+consonant-vowel][+consonant-vowel]/`**. Initial /s/-words (bottom of the screen) have only been expanded to the third position , while non-initial /s/-words (top of the screen) have been fully expanded to their full sequence: /dnɛpr/, /dnjɛstr/ and /pst/

In this graph, nodes represent the states in the finite automaton representation and the links between them the transitions. A node filled with blue, indicates that the node can still be expanded with more children, while a node filled with white indicates that the children already have been expanded. A node can also be filled with a blue circle to indicate that it is an accepting state (i.e. end of word). Transition probabilities are placed either to the left or to the right of the node, depending on it having children or not. Finally, the root node S is the initial, dummy, state.

Being an interactive visualization, users can click on the blue filled nodes to expand the tree. Initially, only the children of the the root dummy state are visualized, and the user decides which path of the chain they wish to pursue. This is a necessary visual feature of the graph, because even visualizing a subtree of the whole phonotactic model can quickly become too large for visualization purposes.

When using this graph to answer the phenomenon of three consonant clusters in Swedish, we see that even though there are in total three outliers in the data, which contradict the stated rule that consonant clusters in Swedish can only start with /s/. These are the two foreign words /dnɛpr/ and /dnjɛstr/ (two long rivers in eastern Europe) and the interjection /pst/ used to imply that the speaker is sending secret information to another person. The large majority of words (more than 99 percent), however, comply to the rule that initial three consonant clusters in Swedish can only begin with /s/.

Implementation

As mentioned above, these visualizations of the phonotactic model have been generated with the visualization library D3.js. Being a JavaScript library, D3.js can be used to create rich, often interactive, visualizations in web browsers (i.e. Google Chrome, Mozilla Firefox or Internet Explorer). Since it incorporates widely implemented web technologies, such as HTML (Hunt, 2010), CSS (Bos, Çelik, Hick-

son, & Lie, 2011) and especially SVG (Dahlström et al., 2011), it has become widely popular within the web development community and is used by media sites, geo-projects and everything in between to create feature rich graphs.

D3.js is essentially a DOM manipulation library (similar to jQuery). The DOM (short for Document Object Model) refers to the logical structure of the web page, including HTML and SVG elements, and D3.js provides the means of selecting, removing, adding and editing these elements in the paradigm of declarative programming. More importantly though, D3.js allows for binding data, whether from CSV, XML, or JSON files to elements in the DOM, thereby making the data drive the visual presentation. To this purpose, the SVG (short for Scalable Vector Graphics) format is most often used, which allows for the creation of geometric shapes, such as circles, rectangles or even more complex shapes like paths. For example, in figure 6, the states of the automaton are mapped to `<circle>` SVG elements and a ring is drawn if the state is an end state. Data-binding can also be done with elements, which are not even present in the DOM. This is done with something called virtual data binding, whereby mismatches between the data and the DOM can be accounted for. For this, there are two important methods, enter() and exit(), where enter() handles a surplus of data in relation to DOM elements and exit() a shortage of data in relation to DOM elements. The visualization in this thesis uses virtual binding extensively to interactively expand or collapse the tree.

Additionally, D3.js API provides many other built-in features to help generate many different kinds of visualizations. These features are meant to assist the core functions of D3.js, however, not to replace them. In this thesis, the built-in tree layout[3] was used, which uses the Reingold–Tilford "tidy" algorithm (Reingold & Tilford, 1981) to calculate what is called tidy trees, which optimally fits the screen in a compact way.

[3]A more extensive description can be found in the documentation on GitHub: https://github.com/d3/d3-hierarchy/blob/master/README.md#tree

4.4. Using Search Patterns to Test Initial Consonant Cluster Rules

With the help of the SEQUENCE(PHONEMES) function and adequate visualization techniques, the research questions relating to Sigurd's (1965) results (question 3) can now be answered. For this thesis, appropriate search patterns have been formulated and their results visualized, in order to investigate the relevant rules laid out by Sigurd (1965) (see section 2.1.5).

For initial two phoneme clusters, search patterns have been formulated, which explicitly contradict Sigurd's rules relating to initial consonant + phoneme permissible sequences. In this thesis, they are called counter-hypotheses, because they define all the initial consonant+phoneme patterns, which directly contradict Sigurd's rules. Furthermore, in order to put possible outliers into a probabilistic perspective, a search pattern, which includes the otherwise excluded phonemes, will also be included.

For initial two phoneme class clusters, search patterns have been formulated to test Sigurd's rules regarding initial consonant phoneme class sequence rules. Here, we will also be able to formulate counter hypotheses but only with those rules that do not have stated exceptions. This is due to the limitation in the current phoneme configuration syntax, wherein specific two-phoneme sequences cannot be explicitly excluded. However, in these cases, we can still answer the questions just by inspecting the output and look for outliers.

5. Results

In this section, I will present the results of the formulated search patterns for the rulesets in Sigurd (1965). This will be done by listing for each rule in the ruleset, 1) the rule as it appears in Sigurd (1965), 2) the relevant search pattern, 3) a description of what the search pattern matches, 4) the visualization of the resulting trie, 5) the words stored in the trie and 6) an analysis of said words. The results will also be summarized in table 6 at the end of this section. What the results entail will be discussed in section 6.

5.1. Initial Two Phoneme Consonant Clusters

Below is the list of the results of the rules for initial consonant+phoneme sequences. For rules which have a consonant exception at the second position (i.e. b), c) an d)), an alternate search pattern and visualization, showing their proportional probability in relation to outliers, are presented. For these, however, only a subset of all matched words will be listed, indicated by three dots (. . .).

a) h, ɕ, ʃj, j, l, r can [only] be followed by a vowel:

Counter-hypothesis: *All words beginning with any of the phonemes /h/, /ɕ/, /ʃj/, /j/, /l/, /r/ and are followed by a consonant.*

Search pattern: `/^[hɕʃjlr][+consonant-vowel]/`

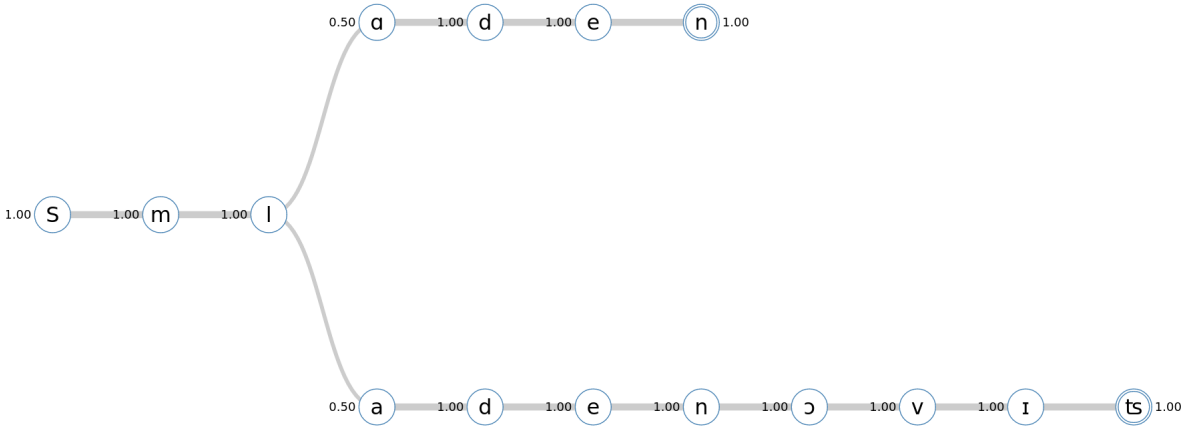
Words: ll, Ljubljana

Analysis: *ll* is probably an, in the lexicon unmarked, acronym for *lättläst* (easy-to-read), while *Ljubljana* is the capital of Slovenia.

b) m, n can [only] be followed by a vowel or by j

Counter-hypothesis: *All words beginning with any of the phonemes /m/, /n/, and are followed by a consonant, which is not /j/.*

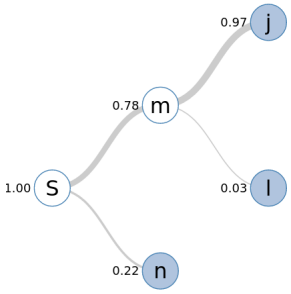
Search pattern: `/^[mn][+consonant-vowel-[j]]/`



Words: Mladenovic (foreign name), Mladen (foreign name)

Analysis: Two foreign names of Slavic origin.

Alternate search pattern: `/^[mn][+consonant-vowel]/`



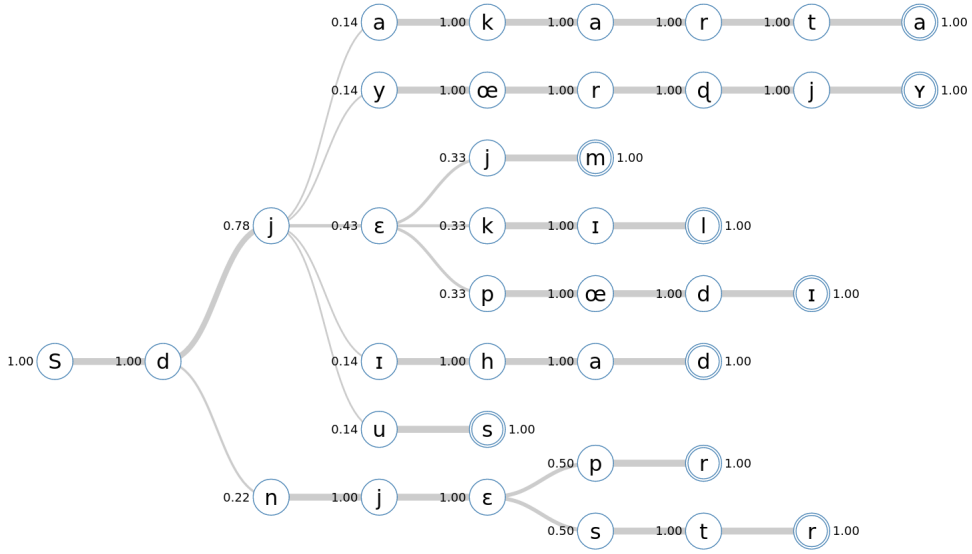
Words: Mjau, Mjukt, Mjölkat . . .

Analysis: The outlier /ml/ phoneme sequence covers only 3% of the cases in the data and the conventional /mj/ sequence 97 %.

c) d, t can [only] be followed by a vowel or by v or r

Counter-hypothesis: *All words beginning with any of the phonemes /d/, /t/ and are followed by a consonant, which is not either /v/ or /r/*

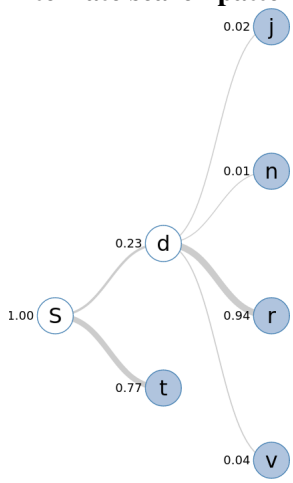
Search pattern: `/^[dt][+consonant-vowel-[vr]]/`



Words: Dnjestr, Dnjepr, Deuce, Jihad, Jeopardy, Jekyll, Jaime, György, Djakarta.

Analysis: Two long rivers in eastern Europe (Dnjestr, Dnjepr), a tennis score of french origin (Deuce), Islamic holy struggle in of Arabic origin (Jihad), game show of English origin (Jeopardy), fictional character of English origin (Jekyll), name of English origin (Jaime), Hungarian name (György), capital of Indonesia (Djakarta).

Alternate search pattern: `/^[dt][+consonant-vowel]/`



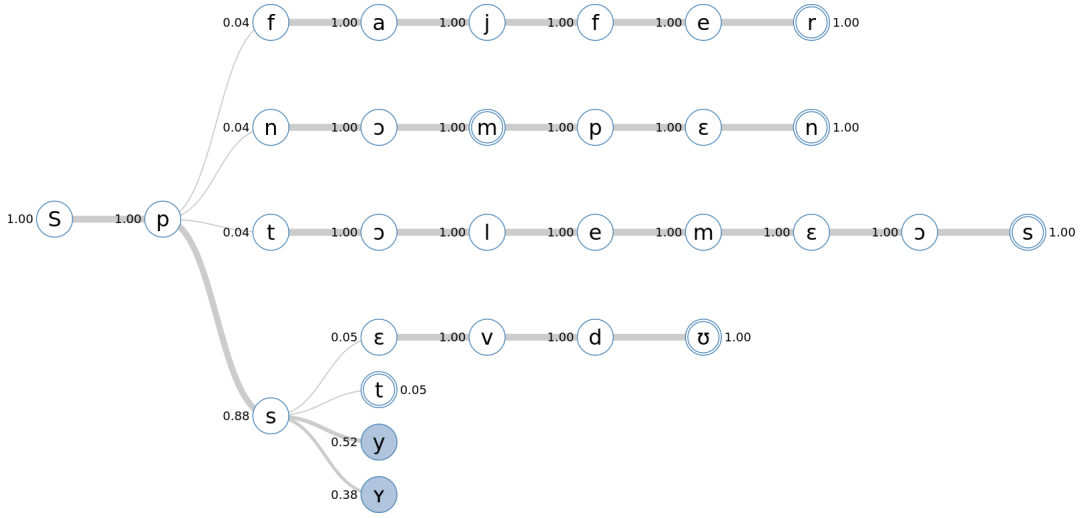
Words: dra, Drake, drag, Dvärggatan, Dvästa, dväljs, tar, tre, ty . . .

Analysis: The outliers /dn/ and /dj/ phoneme sequence respectively covers roughly 3% of the results where /d/ is followed by a consonant. The conventional sequences, /dr/ and /dv/, however, cover more than 97% of the other results, especially /dr/, which accounts for 94% of all results.

d) b, p can [only] be followed by a vowel or by r, l, j

Counter-hypothesis: *All words beginning with any of the phonemes* b, p *and are followed by a consonant, which is which not r, l, or j.*

Search pattern: `/^[bp][+consonant-vowel-[rlj]]/`

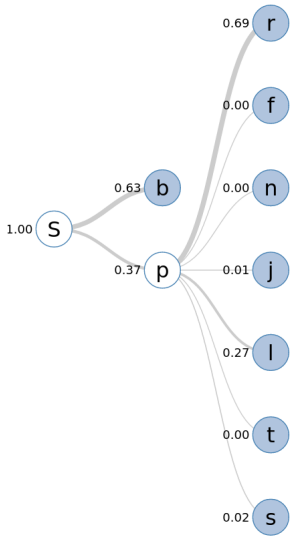


Words: psykologistudierna, psykologiskt, psykologiserade, psykologhjälp, psykiatriutredningen, psykiatrireformen, psykiatrikerkåren, psykedelia, psykvårdsreformen, psykundersökning, psyksjukhus, psyksjukhuset, psykpatienter, psyko-, psykoterapier, psykoneuros, psykklinikerna, psykiskt, psykfall, psykakuten, pst, pseudo-, Ptolemaios, Phnom, Phnom Penh, Pfeiffer.

Analysis: Most words are of the *psyk-* stem, which originally comes from the Greek word *psyché*, which translates roughly to *mind* in English. The transcribers of these words, however, appear to make some sort of excessive pronunciation, seeing that, as Sigurd mentions (Sigurd, 1965, p. 64), most native speakers would substitute /ps/ with /s/, which already is an acceptable phonotactic pattern. The same holds for the word pseudo, which is also of Greek origin. The rest of the words

consist of a Greek astronomer (Ptolemaios, born 90 AD.), the capital of Cambodia (Phonem Penh) and a German surname (Pfeiffer).

Alternate search pattern: `/^[bp][+consonant-vowel]/`



Words: bara, be, Buss, prata, provat, prydligt, plan, platt, plus, Pjäsbacken, pjäxdans, Piero.

Analysis: Although this pattern has returned the most amount of word outliers, the conventional sequences combined, /pr/, /pl/ and /pj/ still make up 97 % of all results and the outliers only 3 %. Interesting to note, however, is that the only non-foreign words of the permissible sequence /pj/, which only make up 1 % of the overall results, is any word with the stem *pjäs-* (play, theatrical performance) and *pjäx* (ski boot, old loan word from Finnish *pieksu*). The rest of the words are all foreign words. An illustration of this sequence can be found in Appendix 1.

5.2. Initial Two Consonant Phoneme Class Clusters

a) stops-stops, (**ptkbdg-ptkbdg**)

Counter-hypothesis: *All words beginning with two stop phonemes.*

Search pattern: `/^[+stop][+stop]/`

Words: Ptolemaios

Analysis: Greek astronomer, born 90 AD.

b) nasals-nasals, (**mn - mn**)

Counter-hypothesis: *All words beginning with two nasals.*

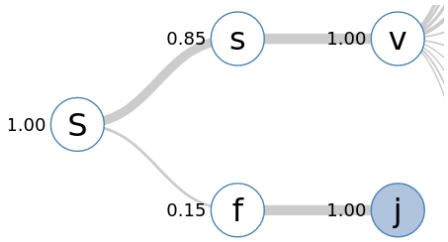
Search pattern: `/^[+nasal][+nasal]/`

Words: None.

c) fricatives-fricatives (**sfvj-sfvj**, exceptions: *sv-fj*)

Counter-hypothesis: *All words beginning with two fricatives.*

Search pattern: `/^[+fricative][+fricative]/`



Words: Only those those starting with sv and fj.

d) liquids-liquids (*lr-lr*)

Counter-hypothesis: *All words beginning with two liquids.*

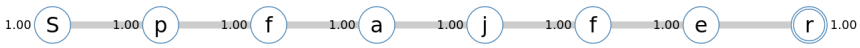
Search pattern: `/^[+liquid][+liquid]/`

Words: None.

e) labials-labials (*pbfmv-pbfmv*)

Counter-hypothesis: All words beginning with two labials.

Search pattern: `/^[+labial][+labial]/`



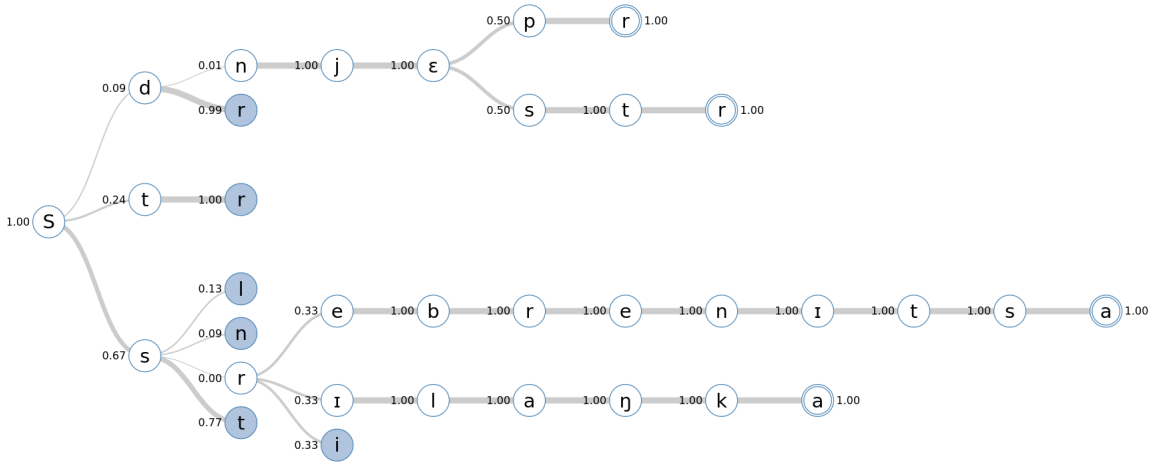
Word: Pfeiffer

Analysis: German surname. /pf/ is a permissible pattern in German.

f) dentals-dentals (*stdnlr-stdnlr*, exceptions: *st, sn, sl, tr, dr*)

Counter-hypothesis: *All words beginning with two dentals.*

Search pattern: `/^[+dental][+dental]/`



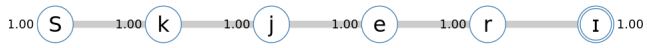
Words: Dnjestr, Dnjepr, Srebrenica, Sri Lanka, Sri Lankas

Analysis: Two long rivers in eastern Europe (Dnjestr, Dnjepr), a town in Bosnia and Herzegovina (Srebrenica), an island country south of India (Sri Lanka and Sri Lankas).

g) palatals-palatals (*kgj-kgj*)

Counter-hypothesis: *All words beginning with two palatals.*

Search pattern: `/^[+palatal][+palatal]/`



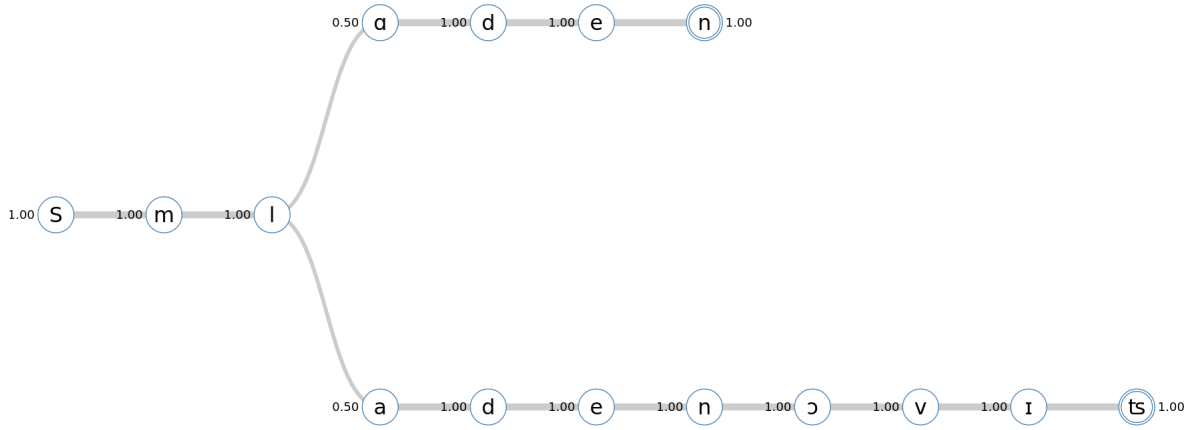
Words: Kieri

Analysis: Finnish surname.

h) nasals-liquids (*mn-lr*)

Counter-hypothesis: *All words beginning with a nasal following a liquid.*

Search pattern: `/^[+nasal][+liquid]/`



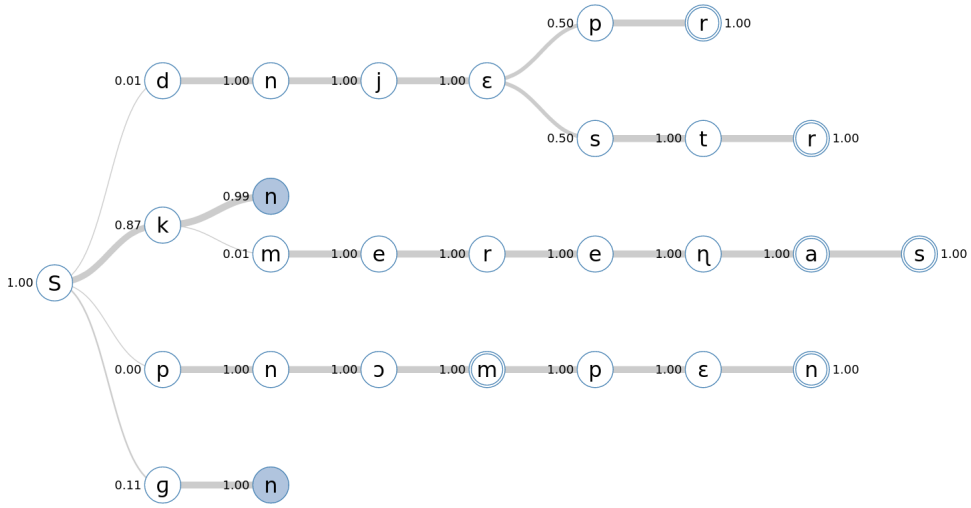
Words: Mladenovic, Mladen

Analysis: Foreign names of Slavic origin.

i) stops-nasals(*ptkbdg-mn*, exceptions: *kn, gn*)

Counter-hypothesis: All words beginning with a stop following a nasal.

Search pattern: `/^[+stop][+nasal]/`



Words: Dnjestr, Dnjepr, Khmererna, Khmerernas, Phnom, Phnom Penh and all words starting with either /kn/ or /gn/

Analysis: Two long rivers in eastern Europe (Dnjestr, Dnjepr), communist party of Cambodia, Khmer Rouge (Khmererna, Khmerernas), the capital of Cambodia (Phnom, Phonm Penh).

5.3. Results Summary

The following table 6 summarizes the results of laid out in previous sections. It does this by stating the rules found in Sigurd (1965), the search patterns used to investigate those rules and the type of results returned by the search pattern.

	Rule	Search Pattern(s)	Result(s)
Consonant Cluster	h, ɕ, ɧ, j, l, r can [only] be followed by a vowel	**/^[hɕɧjlr][+consonant-vowel]/**	Foreign word
	m, n can be followed by a vowel or by *j*	**/^[mn][+consonant-vowel-[j]]/** **/^[mn][+consonant-vowel]/**	Foreign words
	d, t can be followed by a vowel or by *v* or *r*	**/^[dt][+consonant-vowel-[vr]]/** **/^[dt][+consonant-vowel]/**	Foreign words
	b, p can be followed by a vowel or by *r, l, j*	**/^[bp][+consonant-vowel-[rlj]]/** **/^[bp][+consonant-vowel]/**	Foreign words
Class Cluster	stops-stops (*ptkbdg-ptkbdg*)	**/^[+stop][+stop]/**	Foreign word
	nasals-nasals (*mn - mn*)	**/^[+nasal][+nasal]/**	None
	fricatives-fricatives (*sfvj-sfvj*, exceptions: *sv-fj*)	**/^[+fricative][+fricative]/**	Foreign words
	liquids-liquids (*lr-lr*)	**/^[+liquid][+liquid]/**	None
	labials-labials (*pbfmv-pbfmv*)	**/^[+labial][+labial]/**	Foreign words
	dentals-dentals (*stdnlr-stdnlr*, exceptions: *st, sn, sl, tr, dr*)	**/^[+dental][+dental]/**	Foreign words
	palatals-palatals (*kgj-kgj*)	**/^[+palatal][+palatal]/**	Foreign words
	nasals-liquids (*mn-lr*)	**/^[+nasal][+liquid]/**	Foreign words
	stops-nasals (*ptkbdg-mn*, exceptions: *kn, gn*)	**/^[+stop][+nasal]/**	Foreign words

Table 6: A summary of the results of the search patterns, organized by Sigurds rules, the search patterns used to investigate those rules and the results of the search pattern.

6. Discussion

6.1. Method Discussion

The method laid out in this thesis has helped us answer the two first research questions. The answer to the first question, *how can a computational model of Swedish Phonotactics be represented to enable efficient extraction of phonotactic information?*, is that a computational model of Swedish phonotactics can be represented as a finite automaton, expressible by regular languages, allowing for pattern matching against phoneme sequences. This means we can formulate powerful regular expression-like search patterns to extract all kinds of phonotactic information. Although the syntax defined for this thesis is only a small subset of common regular expression standards, much more can be added to it, for example grouping, Boolean quantification and Kleene closure, in order to allow for even more expressive pattern matching. This might prove especially useful for extracting phonotactic information about medial- and final sequences.

The answer to the second research question, *what benefits can a computational model of Swedish phonotactics add to modern Swedish phonotactics?* is that, relating to the first question, efficient pattern matching allows for extracting phonotactic information, which would otherwise require a lot of tedious manual work. Furthermore, by representing the trie as a probabilistic finite automaton and thereby adding probabilities to the phoneme sequences, we can with greater certainty exclude outliers from our general phonotactic description; outliers, which otherwise could only be accounted for by qualitative means. For example, by observing that a specific phoneme combination only occurs with a low proportion of foreign words, we might reach the conclusion that the sequence is too negligible to include in a general phonotactic description of Swedish.

Besides the computational benefits mentioned above, I would argue that the choice for this particular model was important because it complements the fields of study from which it has grown out of. On the one hand, in Swedish phonotactics, because Sigurd's sequence diagram model itself could be represented as a finite automaton, this thesis simply extends on and strengthens that work. On the other hand, in computational phonology, it shows that widely used models within the field can also be used for phonotactic description. In the light of this, the choice of method was well positioned within the areas it grew out of and makes a smooth transition into further research, which is quite practical from a theoretical point of view.

Now, although the model is beneficial for descriptive purposes, it might not completely be suitable for other NLP-related fields. First of all, it does not attempt to be a productive model of phonotactics, i.e one cannot use the model in its current state to generate new permissible phonotactic sequences. This is because a state is defined as a phonemes in a certain position of a (sub-)string of a word, where all sequences are part of specific words and in which end states are always the end of words. Hence, an algorithm, which would randomly walk through the trie, would always end up at the end of a word already present in the lexicon. With a Markov chain representation, however, where phonemes are grouped together on some criterion and transitions occur between these groups, for example on sequence length or syllable boundary, this could be done more efficiently. This is a a common solution for generating random words or texts, where raw corpora is often used, and there is no reason for this not to work with transcriptions as well. Second of all, this model can be seen as a first step in the implementation of other statistical models. For example, in a hidden Markov model aiming to calculate the grapheme of a certain phoneme, known as the grapheme-to-phoneme (g2p) problem, this model could be used as the observed Markov process and graphemes as the hidden states to calculate, in order to model their statistical relationship.

Furthermore, when it comes to visualizing and computing probabilities for medial- and final sequences, the model in its current form has its limitations, since the same phonemes are spread out separately in the tree after the initial position. To solve this for final sequences, the trie would only have to be built backwards starting from the end of the string. For medial sequences, a solution would would be to create a new trie, where initial phonemes are up until the desired positions, so to speak, cut off

from the tree and the new hanging root leaves grouped together. These are implementation changes, which would make the public API more accessible but do not change the formal representation per se.

An important detail left out in this thesis is the question of diacritics and suprasegmentals. Seeing that they play an important role in representing phonemes in IPA, and sometimes even in distinguishing between them, there should be an effort to include them in future versions of either PHONO-STRUCT or other implementations. However, they do present certain practical issues. Firstly, in order to maintain a correct probability distribution, states would have to be re-defined, so that all diacritic versions of phonemes at certain positions are represented as separate states. This increases the amount of nodes in the trie, making it inherently more complex for visualization. A way to solve this would be to separate the internal implementation from the graphical, where states by default are grouped together when visualized and only separated if explicitly specified. When it comes to the suprasegmentals, it is a different story. The way suprasegmentals are defined in IPA they either enclose other phonemes or words, making them non-parsable by regular languages, as we know from the Chomsky hierarchy (Chomsky, 1956). To this purpose, a parser and an appropriate query language would have to be provided, next to the usual pattern matching, in order to match these different computational structures.

Seeing that this model is meant to be used by non-programmers, especially phoneticians, as well as programmers alike, it is important to make the extraction of information and visualization thereof as accessible as possible. Besides the visualization made in this thesis, search patterns should ideally be formulated with the help of a user-friendly user interface. In this sense, Korp, a web interface to extract information from a vast set of corpora [Borin et al., 2012], is a great example. In Korp, a user can formulate advanced queries against the corpus and make all sorts of linguistic investigation. It has, thus, been an important tool for linguists. Having a similar web interface for phonology could, therefore, prove to be of high value.

6.2. Results Discussion

With the results of the formulated search patterns laid out in section 5, we can start answering the third research question, *where do computationally generated phoneme sequences differ from initial two consonant clusters laid out by Sigurd (1965) in terms of phonemes and phoneme classes (distinctive features)?* To begin with, it should be noted that some of the returned outliers are clusters, which were briefly mentioned in Sigurd sigurd1970phonotactic, but not included in the final phonotactics. These include the words of Greek origin /ps/ and /pt/ (e.g. *psykologi, pseudo* and *Ptolemaios*), /sr/ (*Sri Lanka*), and /pf/ from German (*Pfeiffer*). /pt/ was not included, because it only occured as a foreign word, while the other two, /pf/ and especially /ps/, could be substituted with the more familiar phonemes /f/ and /s/, respectively, making these sequences questionable to begin with. Now there were clusters in the results of this thesis, where not mentioned by Sigurd (1965), namely /lj/, /ml/, /dj/, /dn/, /pn/, /kj/, /km/. However, in these cases, we were able to see that all of them only occured with foreign words, while at the same time making up a low proportion of the overall sequences.

To sum it up, I would argue that the results agreed with the rules stated by Sigurd (1965). Even though there were words generated which contradicted Sigurd's rules, they were foreign words, which arguably should not be a part of a general description of Swedish phonotactics. Furthermore, these foreign words were transcribed by foreign pronunciation in mind, which might not be a realistic transcription of Swedish pronunciation, because most native speaker of Swedish would substitute these exotic phoneme combination with a more common Swedish phoneme sequence. Even with non-exotic Swedish words, such as *psykologi*, the transcribers seemed to prefer a more orthographic type of pronunciation, i.e. /ps/ rather than one that more closely represents Swedish phonotactics in speech, i.e. /s/. Additionally, these contradicting sequences tended to have very low frequency in comparison to other phonemes at the same position, indicating that they are just exceptions that prove the rule.

What is important to note about any statistical model, including one of Swedish phonotactics, is that it is a reflection of the data it is fed. In this thesis, a lexicon of largely frequent words, names and places, were used, which might make certain phoneme sequences more emphasized than others. Furthermore,

the method of transcription, i.e. how a transcriber uses IPA symbols to represent speech sounds, might be different from Sigurd's method. To address this, a review of the transcription guidelines of the different projects in NST could be made in order to identify disrepancies between the NST lexicon and Sigurd's work. Nevertheless, the results largely agree with Sigurd's results. Seeing that the conclusions of Sigurd and this thesis come from two arguably different resources, it strengthens the validity of the phonotactic rules studied. However, it is important to note that Sigurd and this thesis have both used pronunciation dictionaries, consisting of only unique words, which does not necessarily reflect phonotactics as it appears in continuous speech. To address this, future projects could use a corpus of transcribed speech to better represent the probability of phoneme sequences in actual speech.

With these results I would argue that it has been demonstrated how beneficial a weighted trie, represented as a probabilistic automaton, of Swedish phonotactics can be. By just formulating simple search patterns, we can easily test hypotheses and give a clear overview of phonemic sequences with the help of visualization. Although the current model does not fully support certain type of analysis, as with final and medial sequences, these can be easily implemented, in order create a full-fledged API for phonotactic purposes. Furthermore, as we have seen, even if transcription data is not perfect, this does not have to be a hindrance, since wrong transcriptions or results contradicting common knowledge within phonotactics, can be detected and possibly explained. Now, this study has only dealt with a small part of Sigurd's work, let alone phonotactics as a field in general, but by using the methodology presented in this thesis or implementing the features mentioned in the previous section, a lot more similar research can be made. This is especially important in terms of final and medial sequences, which were not covered in this thesis. In the end, this thesis is only a proof-of-concept of the fact that this computational model can be used for investigating phonotactic structures, and it is up for future development and research to make it feasible.

Finally, in the light of the results and the methodology presented in this thesis, I would argue that this computational model is more *transparent* and *reliable* than previous methods of working with phonotactics. This is because hypotheses made regarding phonotactic structure in Swedish, can in this model easily be traced back to all empirical data that support it. This differs from Sigurd's results, where the process to find the sequences is not clearly defined, hence inhibiting a replication of the study, and where the amount of example words for sequences are limited to three. This is not to say that the endeavor of Sigurd and other phoneticians have been useless or are not highly important, but that with new technologies their results can be lifted to a higher level of rigorous, evidence-based research.

7. Conclusions

This thesis has investigated a data-driven, computational approach to the modeling of Swedish phonotactics for descriptive purposes. This has been done by defining and implementing Swedish phonotactics as a weighted trie, represented as a proabilistic automaton, where phonemes are linked by transitions in valid phoneme sequences. The model has been populated by a lexicon of Swedish words, including pronunciation in IPA, from Nordisk Språkteknologi.

To aid in this endeavor, three research questions have been formulated. The answers to them are given below:

1. *How can a computational model of Swedish phonotactics be represented to enable efficient extraction of phonotactic information?*

   A model of phonotactics can be modeled as a weighted trie, represented as a probabilistic automaton, which allows for pattern matching with the help of regular expression-like syntax. This gives expressive power to extract all kinds of phonotactic information.

2. *What benefits can a computational model of Swedish phonotactics add to modern Swedish phonotactics?*

   A computational model of Swedish phonotactics, like the one laid out this thesis, adds the benefit of easily extracting phonotactic information, which otherwise would require a large amount of manual work. By implementing the model as a trie, it naturally extends already existing models within Swedish phonotactics, and furthermore supplements them with statistical weighting.

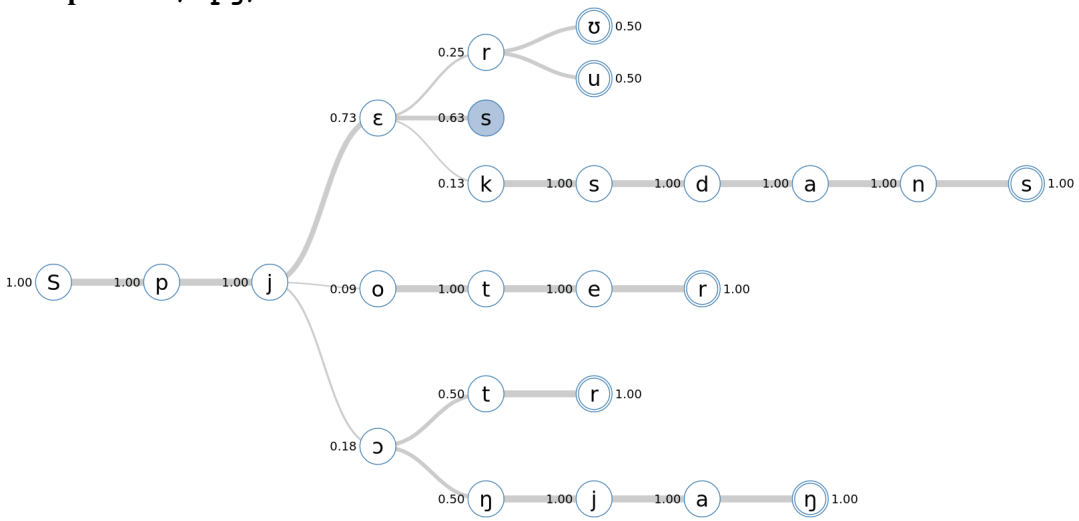
3. *Where do computationally generated phoneme sequences differ from initial two consonant clusters laid out by Sigurd in terms of phonemes and phoneme classes (distinctive features)?*

   Although there were contradicting examples of mostly foreign words, the results largely confirm the long standing phonotactic rules stated by Sigurd regarding two initial consonant clusters of phonemes and phoneme classes.

With these results, it has been proposed that the model laid out in this thesis should be extended to support a complete regular expression syntax as well as diacritics and suprasegmentals in IPA and that the model should, with the help of visualization techniques already laid out in this thesis, be made graphically accessible to programmers and non-programmers alike, for the purposes of researching Swedish phonotactics. With these future endeavors, an important infrastructure, similar to Korp in general linguistics, could be made available for the purposes of studying Swedish phonology and phonotactics.

A. The case of /pj/-

Search pattern: `/^pj/`



Words: Piero, Pierrou, pjäs- . . . , pjäxdans, Piotr, Pjotr, Pyongyang.

References

Akademien, S. (1950). *Saol, svenska akademiens ordlista över svenska språket* (Vol. 9). Author.

Beesley, K. R., & Karttunen, L. (2003). Finite-state morphology: Xerox tools and techniques. *CSLI, Stanford*.

Bird, S. (2002). *Computational phonology* (Tech. Rep.). University of Pennsylvania.

Bos, B., Çelik, T., Hickson, I., & Lie, H. W. (2011, June). *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification* (first Edition of a Recommendation). W3C. (https://www.w3.org/TR/2011/REC-CSS2-20110607/)

Bostock, M., Ogievetsky, V., & Heer, J. (2011). D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*. Retrieved from http://vis.stanford.edu/papers/d3

Brass, P. (2008). *Advanced data structures* (Vol. 1). Cambridge University Press Cambridge.

Catford, J. C. (1988). *A practical introduction to phonetics*. Clarendon Press Oxford.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on information theory*, *2*(3), 113–124.

Clark, J., DeRose, S., et al. (1999). *Xml path language (xpath) version 1.0*.

Dahlström, E., Dengler, P., Grasso, A., Lilley, C., McCormack, C., Schepers, D., ... Jackson, D. (2011, August). *Scalable vector graphics (SVG) 1.1 (second edition)* (second Edition of a Recommendation). W3C. (https://www.w3.org/TR/2011/REC-SVG11-20110816/)

De La Briandais, R. (1959). File searching using variable length keys. In *Papers presented at the the march 3-5, 1959, western joint computer conference* (pp. 295–298).

Ellison, T. M. (1994). *The machine learning of phonological structure*. University of Western Australia.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., & Pallett, D. S. (1993). Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, *93*.

Gasser, M. (1992). Learning distributed representations for syllables. In *Proceedings of the fourteenth annual conference of the cognitive science society* (pp. 396–401).

Group, A. (2008). *The open group base specifications issue 7* (second Edition of a Recommendation). IEEE, The Open Group. (http://pubs.opengroup.org/onlinepubs/9699919799/)

Haugen, E. (1967). A review of phonotactic structures in swedish. *Language*, *43*(3), 803-809. Retrieved from http://www.jstor.org/stable/411820

Hunt, L. (2010, August). *HTML5 Reference* (W3C editor's draft). W3C. (https://dev.w3.org/html5/html-author/)

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

Johnson, C. D. (1970). *Formal aspects of phonological description* (Vol. 3). University of California.

Jurafsky, D., & Martin, J. H. (2000). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition* (1st ed.). Upper Saddle River, NJ, USA: Prentice Hall PTR.

Kaplan, R. M., & Kay, M. (1994). Regular models of phonological rule systems. *Computational linguistics*, *20*(3), 331–378.

Kleene, S. C. (1951). *Representation of events in nerve nets and finite automata* (Tech. Rep.). DTIC Document.

Loman, B. (1967). Synpunkter på svenskans fonotaktiska struktur. i: Arkiv för nordisk filologi 82. *Lund: CWK Gleerup. S*, 1–100.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Nasjonalbiblioteket. (2011). *Swedish lexical database by nordisk språkteknologi holding.* http://www.nb.no/sbfil/dok/nst_leksdat_se.pdf. Author. (Accessed: 2017-05-07)

Reingold, E. M., & Tilford, J. S. (1981). Tidier drawings of trees. *IEEE Transactions on software Engineering*(2), 223–228.

Riad, T. (2013). *The phonology of swedish*. Oxford University Press.

Schone, P., & Jurafsky, D. (2000). Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of the 2nd workshop on learning language in logic and the 4th conference on computational natural language learning-volume 7* (pp. 67–72).

Sigurd, B. (1965). *Phonotactic structures in swedish* (Unpublished doctoral dissertation). Lund universitet.

Unicode Consortium. (1997). *The unicode standard, version 2.0*. Addison-Wesley Longman Publishing Co., Inc.

van Rossum, G. (1995). *Python reference manual* (Tech. Rep.). Amsterdam, The Netherlands, The Netherlands: Python Software Foundation.

Wall, L. (2000). *Programming perl* (3rd ed.; M. Loukides, Ed.). Sebastopol, CA, USA: O'Reilly & Associates, Inc.

Wells, J. C. (1997). Sampa computer readable phonetic alphabet. *Handbook of standards and resources for spoken language systems*, *4*.

Yergeau, T. B. J. P. C. M. S.-M. E. M. F. (2010, November). *Reference* (W3C Recommendation). W3C. (https://www.w3.org/TR/REC-xml/)

Stockholm University
SE-106 91 Stockholm
Telephone 08 - 16 20 00
www.su.se

Stockholms universitet