

# Building a Swedish rhyme dictionary

**Rasmus Arnlung Bååth**

Department of Computer Science  
Faculty of Science, Lund University  
rasmus.baath@gmail.com

**Staffan Åberg**

Department of Computer Science  
Institute of Technology, Lund University  
carnevorum@gmail.com

January 17, 2006

## 1. Abstract

This report describes an attempt to build a Swedish rhyme dictionary. The project is part of the course *Language Processing and Computational Linguistics* given at Lund university the spring term 2005. To be able to rhyme with a word you have to know the pronunciation and so the authors conclude that a phonetic lexicon is needed. Since no comprehensive phonetic lexicon is readily available the first step is to build one out of LEXIN, a small but free lexicon. The second step is to implement a search engine and the requirements of such an engine are discussed.

## 2. Introduction

A rhyme is when two or more words partially sounds the same. There are many categories of rhymes, and what category a rhyme belongs to depends on what part of the words are similar. When there's a similarity in the beginning one speaks of alliteration. Alliteration is common in old Norse writings and here is a famous stanza from the poetic Edda:

“**V**red **v**ar **V**ing-Tor,  
när han **v**aknade,  
och sin **h**ammare,  
**h**an saknade.“

When there's similarity in the middle one speaks of assonance, e.g. *gûta-måla*. The most common type of rhyme is when there's a similarity in the end. This is called a tail rhyme and it's the type of rhyme this report deals with. For the rest of this report a rhyme is always a tail rhyme. Rhymes can be both monosyllabic and polysyllabic, *gris-paris* being the former and *österrikare-trösterikare* being the latter. Rhyming serves two purposes. It gives a text rhythm and flow, hence pop lyrics nearly always rhyme, and it makes a text easier to remember, the fact that nursery rhymes often are very old is proof of this. Rhyming is an old practice and there are examples of rhymed Christian hymns dating back to 300 ad.

Rhyming is not always easy and sometimes you

would want a rhyme dictionary. This report describes an attempt to build one, supposed to be comprehensive, easy to use and accessible over the web. There already exists a number of rhyme dictionaries on the web, but they are all flawed in the same way. They only consider the spelling of a word when searching for rhymes. This is an understandable design choice. It's easy to implement, you just write an algorithm that picks out words from a word list that matches the end of a given word. It's easy to extend, you just add new words to your word list. Though good in many ways it also gives rise to some problems. Even though Swedish spelling is relatively consistent, at least compared to English, there are many exceptions. A rhyme dictionary that only considers spelling would miss perfectly good rhymes such as *fleece-gris* and include *dam-skam*, which doesn't rhyme. When rhyming it is also important consider stress, something that's hard to deduce from the spelling of a word. To cope with such problems one has to consider the pronunciation of the words, not only the spelling. The easiest way to do this is by using a phonetic lexicon when searching for rhymes. One could also attempt to devise rules to derive the pronunciations from the spelling, but this is difficult and a perfect solution is probably AI-complete.

Since no free and comprehensive phonetic lexicon for Swedish is readily available, the task to build a rhyme dictionary is twofold. First one has to get hold of a phonetic lexicon then one has to build a search engine to search it for rhymes. The rest of this document is organized in the following way: Section 3, describes how we take LEXIN, a lexicon aimed at emigrants learning Swedish, and by means of several Perl scripts make it into a phonetic lexicon implemented as a MySQL database. Section 4 discusses the requirements and the implementation of the search engine. Section 5 shortly discusses the interface of the rhyme dictionary. Then follows an evaluation and some concluding comments. Finally there's an appendix consisting of a short explanation of the phonetic alphabet used, an excerpt from LEXIN and a picture of the web interface of the lexicon.

### 3. The Phonetic Lexicon

#### 3.1 . Lexin

Swedish is a small language and it's not strange that Swedish linguistic resources aren't as great as for larger languages, such as English. If you're looking for an English phonetic lexicon you would find Moby Pronunciator<sup>1</sup>, a large and free lexicon consisting of 175,000 words with corresponding pronunciation. A similar lexicon for Swedish doesn't exist. What does exist is LEXIN. LEXIN is a lexicon built by the Institution of Swedish language at the university of Gothenburg and it's free to download from Språkbanken<sup>2</sup>, an online collection of linguistic resources. LEXIN is intended as an aid for emigrants learning Swedish and consists of roughly 20,000 entries. Every entry consists of a lemma, a pronunciation, a part-of-speech tag, a list of inflections, a list of compound words and possibly a description of the usage of the word. LEXIN is stored in the form of an XML document, see appendix A for a short excerpt and appendix B for a description of the phonetic alphabet used.

Before we can use LEXIN as a phonetic lexicon some changes has to be made. First of all LEXIN has to be made into a MySQL database. XML is a good format for structured data, but by using a MySQL database you gain access to powerful tools such as fast search mechanisms and consistency checks. Secondly LEXIN has to be cleaned up. The way the spelling and the pronunciation is notated is inconsistent and the XML document is not valid nor well-formed. There's also a lot of information in LEXIN that we don't need. We don't need to know what part-of-speech a word belongs to or the usage of a word. The phonetic notation is also unnecessarily complicated. Thirdly LEXIN has to be extended. LEXIN's mere 20,000 entries are not enough, for instance the dictionary compiled by the Swedish Academy contains over 120,000 entries.

#### 3.2. Making LEXIN into a MySQL database

To make LEXIN into a MySQL database we use Perl since it has a good database interface and good tools for working with text. First we transform LEXIN into a flat file, this being easier and faster to work with than a XML file. Unnecessary information such as information about what part-of-speech a word belongs to is not included. The flat file is then made into a MySQL database and at the same time parentheses in the spelling and pronunciation of a word, notating alternate spelling or pronunciation, are removed.

1 <http://www.dcs.shef.ac.uk/research/ilash/Moby/>

2 <http://spraakbanken.gu.se/>

The LEXIN database, hereafter only called the database, now consists of four tables. One table consisting of lemma's with corresponding pronunciation, two tables with inflections and compounds, respectively, without corresponding pronunciation and one empty table to hold inflections of compounds.

The database now has to be cleaned up and made more consistent. The notation of words and pronunciations contains elements that we don't need. Eg. “~” that separates parts of multiwords is removed and in the pronunciation “:” after consonants is removed. “:” after a consonant means that the consonant is long and this is not important to consider when rhyming. Monosyllabic words, that for some reason aren't stressed, are made stressed. Inflections can both be complete words, as in [*bagare :: bagaren*], or partially spelled out, as in [*bagare :: -n*]. We want all inflections to be complete words and we therefore complete partially spelled out inflections.

#### 3.3 . Extending the Database

The database now is in the form we want it to be in but it's still not very comprehensive and we would want to extend it. To add completely new words is out of the question since this can't easily be automated. What can be automated is adding pronunciation to the roughly 37,000 inflections. The information we have at our disposal is this: the spelling of the lemma, it's pronunciation and the spelling of it's inflections. We first produces a list of all the differences, hereafter called suffixes, between an inflection and it's lemma. E.g. [*bagare :: bagaren*] would produce “n” and [*fadern :: fäderna*] would produce “äderna”. It turns out that this list consists of under 200 different suffixes and so it's possible to enter their pronunciation by hand. The script for producing the inflections' pronunciations then works in the following way. For each inflection pick out the suffix, the spelling in common with the lemma and the rest of the lemma, e.g. [*driva :: drivor*] would produce (“or”, “driv”, “a”). Then extract the common pronunciation by removing the pronunciation corresponding to the rest of the lemma from the lemma's pronunciation, in this case “*drI:va*” would become “*drI:v*”. Then the pronunciation corresponding to the suffix is concatenated with the common pronunciation, and so “*drI:v*” + “*or*” becomes “*drI:vor*”. This simple method works, but some extra rules are needed. In for example [*dö :: dött*] the *ö* is long in *dö* but short in *dött*, so it's not enough just to add the pronunciation of the suffix “*t*”. This problem can be solved by checking if adding the suffix to the common spelling leads to new double

consonants, if this is the case make the preceding vowel short. Other tricky inflections that has to be dealt with are the ones ending with “*orer*”. E.g. [*dator* :: *datorer*] are pronounced respectively “*2dA:tor*” and “*2da:iO:rer*”. Not only is the “*o*” made long, it is also made stressed.

The method works incredibly well and produces 36,000 new inflection-inflection pronunciation pairs. When checking 100 randomly picked inflection-inflection pronunciation pairs only one was not correct.

Each lemma also has a number of compound words associated with it. The compound words can be divided into two categories, those that ends with it's lemma and those that starts with it's lemma. We can't produce the full pronunciation of the compound words but we can produce a partial by using the lemma's pronunciation. Since our database is going to be used as a rhyme lexicon we are mostly interested in the pronunciation of the end of the words. We therefore only produces partial pronunciation for the compound words that ends with it's lemma. We introduce “*§*” as a wild card sign indicating where we don't know the pronunciation. A compounds pronunciation is then simply it's lemmas pronunciation with a “*§*” added in front, e.g. [*väggalmanacka* :: *§almanaka*] and [*överansträngning* :: *§anstre@ni@*]. This produces 3,500 new word-pronunciation pairs. We make the uppercase letter, indicating stress, into lowercase in all the partial pronunciations since the stress most often lies in the first word of the compound words. We also generate the inflected forms of all the compounds with partial pronunciation producing another 8,000 word-pronunciation pairs. We now have a phonetic database consisting of 63,000 word-pronunciation pairs.

## 4. Searching for rhymes

### 4.1 . Requirements of a search engine.

The strict definition of a rhyme is a word that corresponds with another one from and including the last stressed syllable. A search engine using this definition could work as follows: Accept a word as input, get that word's pronunciation in the database, remove everything from the pronunciation up to the last uppercase vowel and return a list of words which pronunciation ends in the same way as whats left of the search word's pronunciation. But this engine is naïve in many ways and wouldn't suffice. Here's a list of features a search engine should have.

1. You should be able to change the strictness of the search. The strict definition of a rhyme is in many cases to strict. Sometimes, especially

when writing lyrics for a song, the stress doesn't matter that much.

2. The search engine should be able to handle the fact that many words are spelled the same but pronounced differently. E.g. [*banan* :: *2bA:nan*], the one you can safely walk on and [*banan* :: *banA:n*], which you wouldn't want step on.
3. You should be able to search for rhymes for a word not currently in the database. This could be solved by making the search engine accept a phonetic transcription as input, but this approach would make the search engine less user friendly.
4. The search result should be presented in a perspicuous way. Often you now how many syllables the rhyme your looking for should have. A search engine returning a randomly ordered list or a list sorted in lexical order would make it hard to find rhymes with a certain number of syllables.
5. The search engine should be able to handle an incomplete database. Searching for rhymes with a word the database only has partial pronunciation for should still produce a passable result.

### 4.2 .Implementation of the search engine

Our search engine will be implemented in Perl and will use the MySQL database as it's source of information. The search engine is not intended to be used by end users, and will later be made easier to use by means of a web interface. To understand the choices of design we have made one must first know in what way the output is sorted. First of all the output is sorted by how good it matches the search word, that is how many syllables matches the search word's. E.g. if the search word was *analogi*, *astrologi* would be before *byråkrati*. Then the output is sorted by number of syllables and finally in lexical order.

Our search engine will take a word as input and have four customizable options:

`strict_rhyming_matters`, `stress_matters`, `vowel_length_matters` and `table_to_use`. If the first, `strict_rhyming_matters`, is on, the search engine will produce rhymes according to the strict definition of a rhyme. If it's off, any word that has any syllables up to the end of the word pronounced the same way as the search word will be considered to rhyme. If this option is on it would mean that *ananas-kalebass* would be considered a rhyme, even though it is the first syllable of *ananas* that is stressed. The words still has to be stressed the same way, thus *ananas-hölass* wouldn't rhyme. If the second option,

stress\_matters, is off, the search engine won't consider stress. *Ananas-höl~~ass~~* would be considered a rhyme and so would *mak~~aker~~-gröns~~aker~~*. The third option, vowel\_length\_matters, decides whether the vowel length should matter when rhyming. If vowel\_length\_matters is off, word pairs that normally aren't considered rhymes, because of differences in vowel length, are. For example *kanel-pensel* would be considered a rhyme. The last option, table\_to\_use, governs which parts of the database to use when searching for rhymes. By changing this you can for example make the search engine search only for lemmas or inflections.

The search consists of two steps. First a phonetic transcription corresponding to the search word is extracted from the database. Then the phonetic transcription is used to search the database for words that rhyme.

#### 4.2.1 .Getting the phonetic transcription

When trying to extract the phonetic transcription there are two scenarios. First, if the search word is already in the database, the corresponding pronunciations are returned. Secondly, if there's no perfect match, the search engine will try to match with any word that ends with the search word. If there's still no match, the search engine will successively shave away letters from the beginning of the search word and search after words ending with the now shortened search word until there is a match. If the match returns many words the word with the length most similar to the search word is chosen. That word's pronunciation is then shortened to have as many syllables as the now shortened search word. That pronunciation is then returned. An example: The word *mandel~~massa~~* doesn't match anything in the database and so letters are shaved away from the beginning until a match is made. When *mandel~~massa~~* has become *massa* it matches three words in the database. The word with the most similar length is [*pappers~~massa~~ :: 2pAper+smasa*]. Syllables are removed from it's pronunciation to match the number of syllables in *massa* and thus *masa* is returned. Instead of doing this second scenario search one could devise rules that derives the pronunciation from the search word. But since it's the pronunciation of the end of the words that matters when rhyming and we already have a large database containing information about how words sound in the end, this would be a bad approach. Words spelled the same in the end nearly always sound the same in the end.

#### 4.2.2 .Getting the rhymes

Now when we have the pronunciations of the search word we're going to use them to search the database for rhymes. For each of the pronunciations the following is done. If the pronunciation wasn't the result of a perfect match, strict\_rhyming\_matters and stress\_matters are switched off. That's because the stress of the partial pronunciation probably isn't correct. We then search for words with pronunciations that ends like the search word's pronunciation from the first vowel. The search result is then sorted in lexical order and by number of syllables in ascending order.

If the search word's pronunciation is partial we will probably have unwanted rhymes in the search result. E.g. when searching with [*mandel~~massa~~ :: masa*] we will get everything rhyming with *assa* when we really would like to get everything rhyming with *andel~~massa~~*. Our solution to this problem is to remove every word from the search result that doesn't match, from the first vowel, what's left of the search word when we remove from it, from the end, as many syllables as there is in it's pronunciation. In the case of [*mandel~~massa~~ :: masa*] *mandel~~massa~~* is shortened to *mandelm* and every word that doesn't match, from it's first vowel, with *andlem* is removed. This approach is a return to the method to use a words spelling to search for rhymes, but since we only have a partial pronunciation this is the best we can do.

We then remove the first syllable from the search word and remove syllables from the beginning of it's pronunciation until it has equal or less syllables. We then search once more. If strict\_rhyming\_matters is on the search stops when there's no stressed syllable left in the search words pronunciation. Else we continue searching until there's no syllables left in the search word. The result of the searches are consecutively added to a list. When the search is finished we have a list of rhymes sorted by number of matching syllables in descending order, number of syllables in ascending order and lexical order. Finally the search word and duplicate words are removed from the list .

This method of search seems to work pretty well, though a problem is that it's not very fast. When searching with a search word that already is in the database, the database is accessed about 3-6 times. When searching with a search word that's not in the database, the database could be accessed as many times as the number of letters in the search word. Accessing the database is slow and therefor the search method is slow.

The strategy to use the spelling to weed out rhymes when the search word only has a partial

pronunciation can be discussed. There are many cases when a word isn't pronounced as it's spelled but in contrast to only using the spelling to search for rhymes we here don't use it to match the end of the word. There we use the partial pronunciation. This approach will never remove a correct rhyme, the rhyme will only occur in the wrong place in the search result.

## 5. The Interface

We now has a working rhyme dictionary but it's not accessible nor very user friendly. To use it one would have to set up ones own MySQL database spending a long time building it by running various perl scripts. Then one have to learn rudimentary perl syntax to use it, something nobody should be forced to learn. Therefore we made a web interface (see appendix C). The interface is a HTML form with an underlying cgi script that handle the search queries. The form consists of a field for entering text and two options. The text field is of course used for entering the word to rhyme with. The first option governs the strictness of the search, that is it governs which of the three options `strict_rhyming_matters`, `stress_matters` and `vowel_length_matters` that should be switched on. Here follows the different setups of the four levels of strictness :

```
strikt = (true, true, true)
normal = (false, true, true)
utan betoning = (false, false, true)
nödrim = (false, false, false)
```

We could of course have allowed the three options to be changed directly, but we feel that our approach is more intuitive. The other option governs what database to use when searching for rhymes. The user can choose between *standard* and *utökad*. *Standard* contains all the words which pronunciations are complete, that is all the original lemmas and their inflections. *Utökad* is like *standard* plus all the words with partial pronunciation, that is the compound words and their inflections. The *Utökad* database also contains words submitted through the homepage.

The search result is presented in columns, one for every number of syllables that was matched. If a search is made with the strictness level *normal* it turns out that the column containing words that only matched one syllable contains far to many words to be easily browsed. The use of columns allows the user to first check the column with the "best" rhymes and then, if necessary dive into the other "worse" columns.

The homepage also contains a form that allows

the user to add new words to the database. At the moment users can't change words already in the database. This functionality should be added in the future so that users could fully participate in extending and correcting the lexicon.

## 6. Evaluation

We started this report by complaining about that existing on-line lexicons uses spelling and not pronunciation to search for rhymes. We argued that the former approach misses many rhymes and includes words that doesn't rhyme. The question one ask oneself now is; how good does our rhyme dictionary perform in comparison other dictionaries. Though it's hard to find anything to measure some things could be pointed out.

The fact that our database only consists of 63,000 entries cripples our dictionary. 63,000 may sound a lot but one should remember that most of these are inflected forms of the 20,000 lemmas. Other on line rhyme dictionaries, e.g. DbLex<sup>3</sup>, have databases with over 400 000 words. Our dictionary also suffers from some blind spots, e.g. our database contains no names of geographical locations. Where our lexicon excel is when the *strikt* search mode is used. A *strikt* search produces only rhymes according to the strict definition of a rhyme, that is rhymes that rhyme perfectly. No other rhyme dictionary on the web is able to do something similar.

## 7. Conclusion

This project has led to the creation of a working rhyme dictionary that uses the pronunciation of a word when searching for rhymes. This makes our dictionary unique (we believe) since all other Swedish rhyme dictionaries uses the spelling of a word when searching for rhymes. The dictionary's phonetic database was created out of Lexin, a small but free Swedish dictionary. Considerable amounts of time have been spent transforming Lexin into a usable database. This was not anticipated when the project started. Even though Lexin isn't very comprehensive we feel that our dictionary is usable and that it perform well in comparison to other existing dictionaries.

One thing that could be done to improve our dictionary further is to make it into a proper wiki. This could help solving the problem with our insufficient database. At the moment it's just possible to enter new words, not alter old ones.

Our dictionary is currently not accessible over the Internet since we haven't found anywhere to put it. The license of Lexin may also prohibit us from making our dictionary accessible over the

---

3 <http://www.dblex.com/>

Internet. If our dictionary finds somewhere to live a link will probably be found on the projects course homepage, <http://www.cs.lth.se/DAT171/>. The source code of the project and directions on how to set up your own rhyme dictionary should also be available from there.

## **8. Acknowledgments**

We wish to thank Pierre Nugues for his support during this project.

## **9. References**

Svenska ord/LEXIN vid Språkbanken, Göteborgs universitet. <http://spraakbanken.gu.se/>

Fredrik Hansson and Lennart Nilsson. 1996. *Rimlexikon*. ICA Bokförlag.

Bengt Sigurd. 1991. *Språk och språkforskning*. Studentlitteratur.

Staffan Bergsten. 2002. *Rim & reson*. Nationalencyklopedin. [www.ne.se](http://www.ne.se).

## Appendix A

### Lexin's DTD:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!--                                     -->
<!--          LEXIN                       -->
<!--          Svenska ord, 2:a uppl.      -->
<!--          Copyright Språkdata, Göteborgs universitet -->
<!--          XML Markup: Yvonne Cederholm, Susanne Mankner -->

<!DOCTYPE lexin [
<!ELEMENT lexin (lemma-entry)>

<!ELEMENT lemma-entry      (form, pronunciation, inflection, pos, lexeme*)>

<!ELEMENT form             (#PCDATA)>
<!ELEMENT pronunciation   (#PCDATA)>
<!ELEMENT inflection      (#PCDATA)>
<!ELEMENT pos             (#PCDATA)>
<!ELEMENT lexeme          (lexnr?, definition?, usage?, comment?,
                           valency?, grammat_comm?, definition_comm?,
                           example*, idiom*, compound*)>

<!ELEMENT lexnr           (#PCDATA)>
<!ELEMENT definition      (#PCDATA)>
<!ELEMENT usage           (#PCDATA)>
<!ELEMENT comment         (#PCDATA)>
<!ELEMENT valency         (#PCDATA)>
<!ELEMENT grammat_comm    (#PCDATA)>
<!ELEMENT definition_comm (#PCDATA)>
<!ELEMENT example         (#PCDATA)>
<!ELEMENT idiom           (#PCDATA)>
<!ELEMENT compound        (#PCDATA)>
```

### A sample of Lexin:

```
<lemma-entry>
  <form>geometri</form>
  <pronunciation>jeometri:</pronunciation>
  <inflection>geometri(e)n</inflection>
  <pos>subst.</pos>
  <lexeme>
    <definition>vetenskapen om de matematiska rumsstorheterna</definition>
  </lexeme>
</lemma-entry>
<lemma-entry>
  <form>ger med sig</form>
  <pronunciation>je:rmE:(d)sej</pronunciation>
  <inflection>gav gett (el. givit) ge(!)</inflection>
  <pos>verb</pos>
  <lexeme>
    <definition>acceptera något (efter påtryckning), foga sig</definition>
    <valency>A &amp;</valency>
  </lexeme>
</lemma-entry>
<lemma-entry>
  <form>gerilla</form>
  <pronunciation>2gerIl:a</pronunciation>
  <inflection>gerillan gerillor</inflection>
  <pos>subst.</pos>
  <lexeme>
    <definition>motståndsrörelse, icke reguljära trupper</definition>
    <compound>gerilla~soldat -en</compound>
    <compound>gerilla~verksamhet -en</compound>
  </lexeme>
</lemma-entry>
<lemma-entry>
  <form>gest</form>
  <pronunciation>$es:t</pronunciation>
  <inflection>gesten gester</inflection>
  <pos>subst.</pos>
  <lexeme>
    <definition>(hand) rörelse, åtbörd</definition>
    <usage>bildligt "handling avsedd att visa en persons känslor etc"</usage>
    <example>livliga gester</example>
    <example>en tom gest</example>
    <compound>försoningsgest</compound>
  </lexeme>
</lemma-entry>
```

## Appendix B

Here follows a brief description of the phonetics used in our dictionary.

The following consonants have the pronunciation usually associated with them:

[*b, d, f, g, h, j, k, l, m, n, p, r, s, t, v*]

Consonants missing are: [*c, q, w, x, z*]

The pronunciation of these consonants can be expressed using the consonants above.

E.g. [*cirkus :: sIrkus*], [*yxa :: Yksa*] and [*zoo :: sO:*].

The following vowels have the pronunciation usually associated with them:

[*a, o, u, å, e, i, y, ä, ö*]

A vowel can be long or short. A [*:*] after a vowel indicates the former, a vowel without [*:*] indicates the latter. E.g. [*ful :: fU:l*] and [*full :: fUl*].

There are three symbols that represent pronunciation not usually associated with them: [*c, \$, @*].

[*c*] represent the way the *k* sounds in **kär**. [*\$*] represent the way the *stj* sounds in **stjärna**. [*@*] represent the way the *ng* sounds in **springer**.

[*2*] in the beginning of a word indicates grave accent.

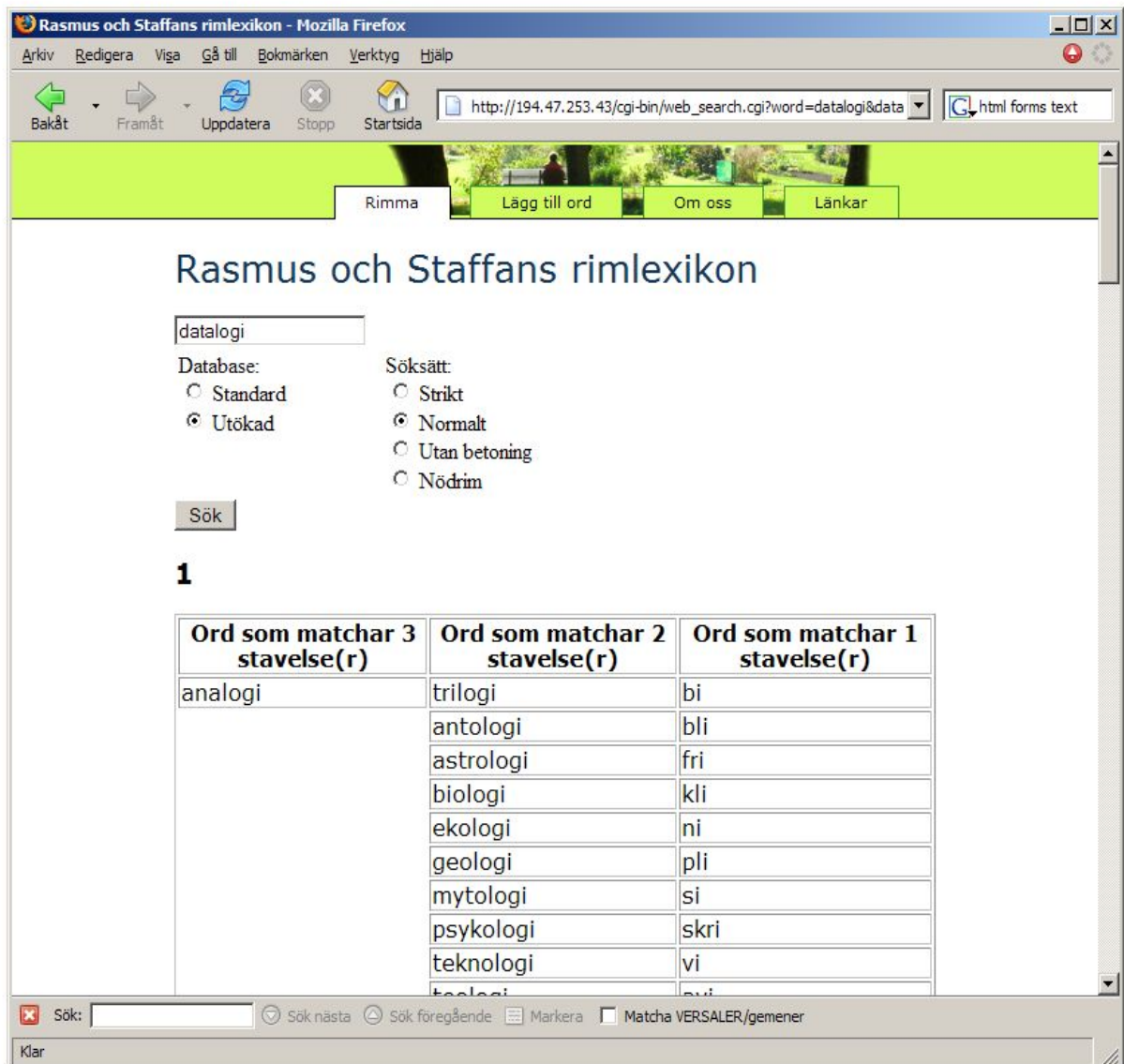
[*+*] between two symbols indicates that the two symbols pronunciation is melted together. E.g. between two vowel this would indicate a diphthong.

What syllable in a word that is stressed is indicated by a capital vowel.

These phonetics are nearly identical to the phonetics used by Lexin, with one major exception. In Lexin [*:*] could also be used after consonants to indicate long pronunciation.



## Appendix C



The screenshot shows a Mozilla Firefox browser window with the address bar containing the URL `http://194.47.253.43/cgi-bin/web_search.cgi?word=datalogi&data`. The page title is "Rasmus och Staffans rimlexikon". The search term "datalogi" is entered in the search box. The search options are set to "Utökad" for the database and "Normalt" for the search method. The search button is labeled "Sök".

The search results are displayed in a table with three columns: "Ord som matchar 3 stavelse(r)", "Ord som matchar 2 stavelse(r)", and "Ord som matchar 1 stavelse(r)". The results are as follows:

Ord som matchar 3 stavelse(r)	Ord som matchar 2 stavelse(r)	Ord som matchar 1 stavelse(r)
analogi	trilogi	bi
	antologi	bli
	astrologi	fri
	biologi	kli
	ekologi	ni
	geologi	pli
	mytologi	si
	psykologi	skri
	teknologi	vi
	teologi	vi

The browser's status bar at the bottom shows "Sök:" followed by a search box, "Sök nästa", "Sök föregående", "Markera", and a checkbox for "Matcha VERSALER/gemener". The status bar also displays "Klar".

*The result of a search using our rhyme dictionary.*