

The Lexical Database of The Bank of Finnish

Mickel Grönroos

**Technical Reports, No. TR-3
Department of General Linguistics
University of Helsinki
September 2000**

The Lexical Database of The Bank of Finnish

Mickel Grönroos
(Mickel.Gronroos@helsinki.fi)

Department of General Linguistics
P.O. Box 4 (Vuorikatu 4 A 7)
FI-00014 University of Helsinki
Finland

Technical Reports, No. TR-3
Department of General Linguistics, University of Helsinki
September 2000

ISSN 1238-6995 ISBN 951-45-9563-7

Contents

Abstract	4
1 Putting language into a database: SKTP-LT	5
2 Using SKTP-LT with the corpus query tool Lemmie	8
2.1 Some notes on usability	8
2.2 Lexical Lookup	9
2.3 Fetching KWIC-lines	10
2.4 Forcing a collocate	11
2.5 Getting more context	12
2.6 Fetching collocations	12
2.7 Fetching colligations	13
2.8 An example	13
3 Some possible expansions of Lemmie and the lexical database	16
References	18
Appendix A. Getting started with Lemmie	
Appendix B. A brief note on regular expressions in Lemmie	
Appendix C. Abbreviations	
Appendix D. The Finnish tag set in SKTP	
Appendix E. The texts in SKTP	

Tables and screenshots

Table 1. The wordform "kutsumme" in TS 1998 in SKTP-LT	6
Table 2. All available inflections of the noun "kutsu" in TS 1998	7
Screenshot 1. The main window at start-up	9
Screenshot 2. An example of the use of Lemmie	14
Screenshot 3. Lemmas collocating with "hiili" in any inflection	15

Abstract

The **Bank of Finnish** (*Suomen kielen tekstipankki*, SKTP) is growing rapidly with texts of different genres and text types. Currently (in September 2000), it contains about 175 million running words in TEI-conformant SGML-markup (see Sperberg-McQueen & Burnard 1994). Approximately two thirds of the texts have been tagged with morphosyntactic data. The corpus consists of newspaper texts, articles from periodicals, novels and nonfictional books.

The material is mainly in Finnish, but a part of the collection is in Finland-Swedish.

The measures taken to cope with such a large amount of text are presented in this paper. The paper consists of two parts.

The first part introduces the **Lexical database of the Bank of Finnish** (*Suomen kielen tekstipankin leksikaalinen tietokanta*, SKTP-LT). The main argument in this part of the paper is the importance of offering the baseform and a variety of grammatical patterns as search keys when studying Finnish text.

In the second part I present the graphical corpus query tool **Lemmie**, which allows baseforms as well as several grammatical features to be used as search criteria.

1 Putting language into a database: SKTP-LT

Most of the linguists who work with language corpora agree on that many types of linguistic information can only be found in corpora of extensive size.

One of the disadvantages of very large corpora (e.g. a million files containing a few hundred million running words) is that the programs you run on your corpus files will take a considerable time to execute.

What one needs is a robust method for speeding up the process of retrieving data.

Typically, an index is generated in which all unique tokens (i.e. types) of the corpus are individually pointed out (e.g. in byte offset counts), thus making them possible to fetch in the blink of an eye. This is a widely used technique and it works very well, primarily for simple lookups on single tokens, but also for phrasal searches etc.

Using indexes is a practical solution when you are working with a corpus consisting of raw text, possibly with some structural markup. However, with the growing number of accurate and fast lemmatizers and taggers available for several languages, it would not be sensible to leave out information provided by software of this type—i.e. baseforms and morphosyntactic descriptions—from the corpus and the corpus query programs. Moreover, the interaction between the different aspects of the lexical entries can be handled in a relational database.

Offering grammatical patterns and especially baseforms to the user as possible search criteria is particularly desirable when working with a mostly agglutinating language like Finnish, where the possibility of using the baseform as search key is demonstrably important. Consider trying to find all available inflections of some Finnish verb in a corpus yourself, when you know there is a theoretical chance that you will have to test for the presence of thousands and yet thousands of different inflected forms (see Karlsson 1986, p. 20). Obviously, you save a lot of time just by knowing that the baseform of a verb can be used to retrieve all the inflected forms of it available in a corpus.

Taking these arguments into account, I decided to develop not only a simple index, but an easily extendable lexical database, allowing several techniques to be used for both datadriven and databased corpus studies.

The solution I have implemented is to store away the morphosyntactic and lemma information found in the corpus in a relational database (in MySQL, see <http://www.mysql.org/>) in conjunction with the appropriate pointer information needed for each and every token. Using this approach, every type generates a lexical entry of its own including information about baseform and part-of-speech (POS) as well as a morphosyntactic description (MSD). Naturally, all the tokens of a type can still be fetched rapidly if needed. And, as stressed above, the database itself becomes a source of lexical information. (See examples in tables 1 and 2.)

In SKTP-LT the following information is gathered and saved for a wordform X in addition to the pointer information:

- X's morphosyntactic features, e.g. Noun Gen SG
- The overall frequency of X in the subcorpus in question
- The amount of different corpus files that X is present in (roughly analogous to the amount of newspaper articles, book pages etc. that X occurs in)
- The baseform Y of X
- The combined frequency of all available inflected forms of the baseform Y
- The amount of available inflections of Y

The word "kutsumme" may serve as an example. Searching for this string in *Turun Sanomat* 1998, which is one of the newspapers materials available in SKTP-LT, will result in the following output:

Table 1. The wordform "kutsumme" in *Turun Sanomat* 1998 in SKTP-LT

type	f(t)	files	lemma	f(l)	inflcs	pos	msd
kutsumme	1	1	kutsu	325	30	Noun	Genom SG P 1P
kutsumme	14	13	kutsua	2037	104	Verb	Pr Act Ind P 1P

Here we see that "kutsumme" as a noun ("our invitation") is much less frequent (1 occurrence) than the verbal reading of the same string ("we are inviting", 14 occurrences). Needless to say the noun "kutsumme" is found in just one corpus file (which is more or less analogous to one article in SKTP), whereas the verbal reading is found in 13 different files. The latter gives us some idea about the usability of the wordform as a verb; as almost every occurrence is found in a different file, we can be quite sure that this is not a word that is specific for just one writer.

Let us study the database output in more detail. It seems that the baseform of the noun, "kutsu", occurs in 30 different inflections with a combined frequency of 325 tokens.

Well, which are these inflections, then? Fetching them is no problem (here they are ordered alphabetically, though ordering by descending frequency is naturally possible and easily done as well):

Table 2. All available inflections of the noun "kutsu" in *Turun Sanomat* 1998 in SKTP-LT

type	f(t)	files	lemma	f(l)	inflcs	pos	msd
kutsua	57	54	kutsu	325	30	Noun	Part SG
kutsuakaan	1	1	kutsu	325	30	Noun	Part SG
kutsuhan	1	1	kutsu	325	30	Noun	Nom SG
kutsuihin	1	1	kutsu	325	30	Noun	Ill PL
kutsuilla	7	7	kutsu	325	30	Noun	Ad PL
kutsuillaan	1	1	kutsu	325	30	Noun	Ad PL 3P
kutsuille	2	2	kutsu	325	30	Noun	All PL
kutsuillesi	1	1	kutsu	325	30	Noun	All PL S 2P
kutsuilta	1	1	kutsu	325	30	Noun	Abl PL
kutsuineen	1	1	kutsu	325	30	Noun	Com PL 3P
kutsuinkin	1	1	kutsu	325	30	Noun	Instr PL
kutsuista	5	5	kutsu	325	30	Noun	El PL
kutsuja	15	14	kutsu	325	30	Noun	Part PL
kutsujakin	1	1	kutsu	325	30	Noun	Part PL
kutsujen	8	8	kutsu	325	30	Noun	Gen PL
kutsuksi	1	1	kutsu	325	30	Noun	Transl SG
kutsulla	1	1	kutsu	325	30	Noun	Ad SG
kutsulta	1	1	kutsu	325	30	Noun	Abl SG
kutsumme	1	1	kutsu	325	30	Noun	Genom SG P 1P
kutsun	119	112	kutsu	325	30	Noun	Gen SG
kutsuna	1	1	kutsu	325	30	Noun	Ess SG
kutsunkin	1	1	kutsu	325	30	Noun	Gen SG
kutsunsa	2	2	kutsu	325	30	Noun	Genom SG 3P
kutsussa	4	4	kutsu	325	30	Noun	In SG
kutsussaan	1	1	kutsu	325	30	Noun	In SG 3P
kutsusta	53	53	kutsu	325	30	Noun	El SG
kutsustaan	2	2	kutsu	325	30	Noun	El SG 3P
kutsut	18	16	kutsu	325	30	Noun	Nom PL
kutsutta	1	1	kutsu	325	30	Noun	Ab SG
kutsuun	16	12	kutsu	325	30	Noun	Ill SG

Note that this information is retrieved through the baseform "kutsu" in combination with the restriction "Noun", not through the individual wordforms. Consequently, the user is not forced to fetch all the inflections separately and thus does not have to try any inflections that are not actually present in *Turun Sanomat* 1998.

2 Using SKTP-LT with the corpus query tool Lemmie

2.1 Some notes on usability

The SKTP user uses the lexical database mainly through the graphical corpus query tool, called *Lemmie*. However, *Lemmie* is not only a graphical user interface (GUI) for SKTP-LT, but also a concordancer and collocator for the Bank of Finnish. One can say that *Lemmie* links the database to the corpus files.

When developing the current version of *Lemmie* (1.0), there were a few conditions that had to be met. In this section I will first present these conditions and then more specifically study how *Lemmie* works.

The first criterion is that it should be *easy* to query SKTP-LT. The regular user has to be able to do the queries he wants to do, without having to know any SQL¹ at all.

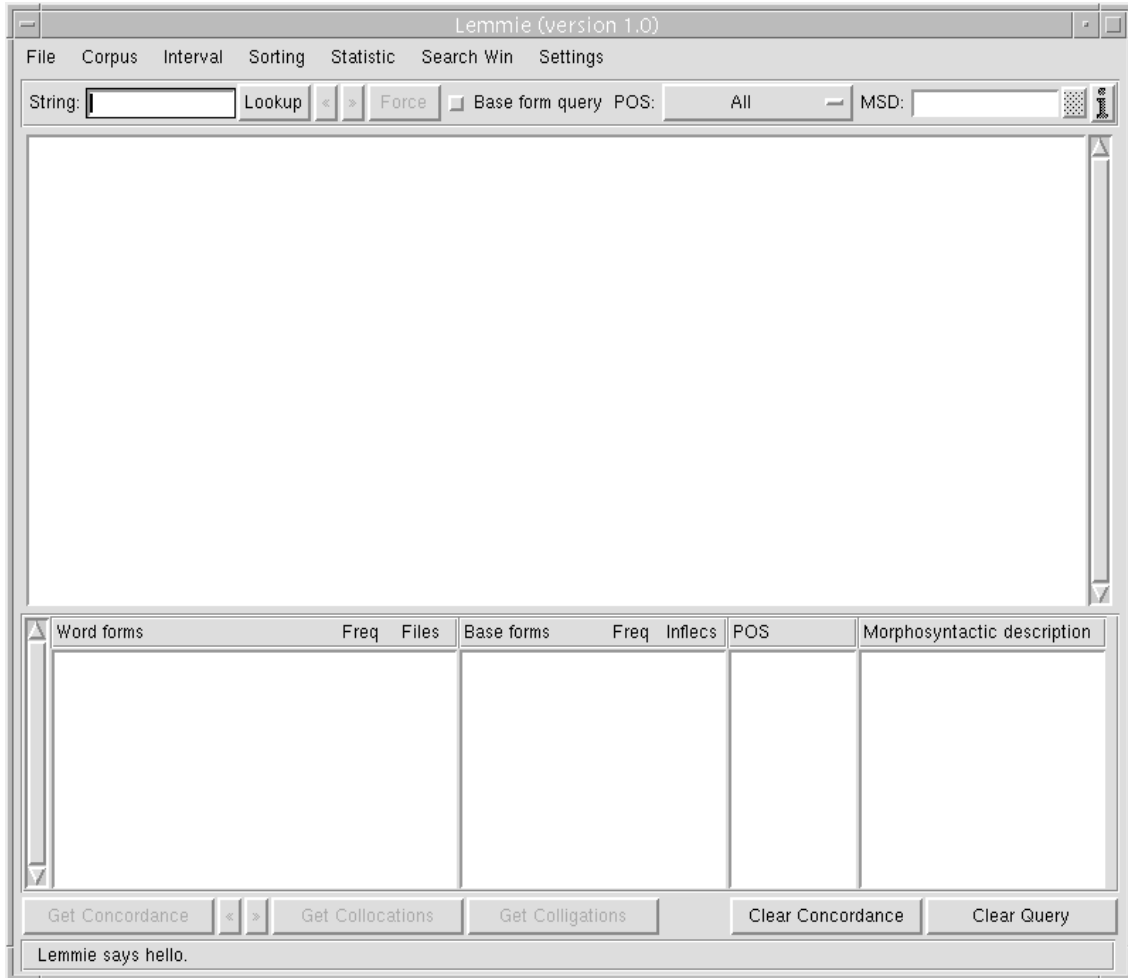
The second criterion is that *Lemmie* should provide the user with the basic service one would expect any corpus linguist might need. In other words, *Lemmie* does not write your Ph.D. thesis—it probably does not even cover half of the things you need for that—but it should at least be able to perform the basic tasks for the user, so that he can concentrate on the aspects of language he is really interested in.

The third criterion is that the connection between the database and the context—in forms such as a KWIC-concordance or a collocational or colligational table—should be smooth. The user should not have to worry about what happens internally inside the computer when he presses some button.

The fourth and last criterion is that *Lemmie* should be fast. This and the third criterion actually do collide sometimes, but I have chosen to let that happen; if somebody really wants to fetch all verbs from the database or browse through all KWIC-lines with, for instance, "ja" (Conjunction, "and") as node, then he should be able to do so. Searches with high coverage like these have a price, though, in long execution time or simply a refusal from the MySQL server to complete the query. Anyway, the issue here is that the less exceptional searches still work at a reasonably good pace to meet the forth criterion.

Let us move on to see how *Lemmie* works.

¹ SQL = Structured Query Language, a standardized query language for requesting data from a database.

Screenshot 1. The main window at start-up.

All queries passed to *Lemmie* go through the lexical database and the query is initiated when the user presses the *Lookup*-button (see screenshot 1).

Before pressing this button the user must naturally specify the combinations of search criteria that are to be met. These are entered by typing something into the *String*-field, by selecting or deselecting the *Base form query*-checkbox, by choosing a part-of-speech (POS) the words must match or by entering a morphosyntactic description (MSD) in the *MSD*-field. The user is free to combine all search criteria in any way.

The *String*-field accepts the usual truncations, where '*' (asterisk) means zero or more characters and '.' (dot) means one arbitrary character. Examples: "hiili*" ("hiili", "hiilialue", "hiilidioksidi", "hiiligrilli", "hiilihappo", "hiilikaivos"), "*hiili" ("hiili", "kivihiili", "riikkihiili", "ruskoshiili") and "h.ili" ("haili", "heili", "hiili", "huili")².

² Full support for regular expressions is *not* provided in the *String*-field during lexical lookup. However, regular expressions can be used in the *String*-field when forcing a collocate. More on this subject later on.

If the user checks the *Base form query*-checkbox, all queries will be done on baseforms (lemmas) instead of wordforms. If this box is checked, a query with "puhaltaa" ("to blow", Verb) as search string will ask *Lemmie* to retrieve all possible inflections (and possibly derivations) that it can find in the specified subcorpus for that lemma. In the *Turun Sanomat* texts from 1998 that would summon up to 56 different wordforms: "puhalla", "puhalleta", "puhalletaan", ..., "puhaltavine", "puhaltavista".

In the *MSD*-field any morphosyntactic descriptions or parts thereof may be entered. Regular expressions are permitted, as in "^Nom SG" and "I?Ipartic\$". (Here the circumflex, '^', matches the beginning of the description and the dollar sign, '\$', matches the end. The question mark, '?', means that the previous character, the capital 'I', is optional. See appendix B for a brief note on regular expressions).

When the user is ready to fetch the matching words from the database, he presses the *Lookup*-button (or simply hits the return key).

Lemmie will now fetch and show the matching wordforms and information about their frequency in the specified subcorpus. The number of different files the wordforms occur in will be shown, as well as the baseforms of the matching wordforms, the joined frequency of the different inflections/derivations of the baseform, the amount of inflections/derivations available in the subcorpus and part-of-speech and morphosyntactic data for the wordforms when available.

If there are more matching words than the number specified in the *Interval*-menu, one can scroll to the next chunk of matching words by clicking the button marked '»' to the right of the *Lookup*-button. Pressing '«' will make *Lemmie* scroll back to possible previous words matching the query.

Given that the user has made a successful lookup in SKTP-LT, *Lemmie* offers four different ways to proceed. The user can ask *Lemmie* to

- Do another lexical lookup
- Fetch KWIC-lines
- Calculate collocations
- Calculate colligations

As the lexical lookup was covered in this section, I will move directly to the fetching of KWIC-lines.

2.3 Fetching KWIC-lines

When the user has made a successful database lookup, three buttons become clickable in the lower left corner of *Lemmie*'s main window. These are *Get Concordance*, *Get Collocations* and *Get Colligations*.

A KWIC-line is a line of text with a specified keyword in the middle, in other words a "keyword in context" (hence the abbreviation). A set of these lines are called a concordance.

To fetch a KWIC-concordance in *Lemmie*, you need to specify what you want to use as keyword. This is done by selecting a wordform, baseform (which will

make all available inflections/derivations keywords as well) or morphosyntactic description in the boxes containing the results of the lexical lookup. Note that it is not possible to choose a part-of-speech tag only as keyword, as this presumably would generate too many lines, thus making *Lemmie* too slow.

To fetch the KWIC-lines using the selected wordform, baseform or morphosyntactic description (MSD) as node, the user has to press the *Get Concordance*-button (or double-click the selected item). *Lemmie* will now build a KWIC-concordance around the selected node. This might take some time if there are a lot of matching keywords, due to the fact that *Lemmie* sorts the concordance before showing it to you. Therefore the maximum amount of lines actually read and processed can be changed (the default is 1000), so if the user wants the process to be fast and he does not bother about getting a lot of KWIC-lines, setting this value low will speed up the process considerably. Setting the value high is recommended if the user also intends to use a forcing collocate to find phrases or multi-word patterns at a later stage.

2.4 Forcing a collocate

As we have seen so far, the KWIC-concordance is built around one keyword, which can be a wordform, baseform or morphosyntactic description (including part-of-speech information).

However, there are situations where one would want to use several keywords as node, e.g. if one wanted to fetch KWIC-lines of certain multi-word patterns, phrases, collocations or colligations. This is where the forcing collocate is used.

A collocation is a combination of words, that occur more frequently than one would expect given the individual frequencies of the tokens (see Firth 1957, p. 194, Smadja 1993, pp. 143—189 or Butler 1998, pp. 1—2). Consequently, by "forcing a collocate" I mean narrowing down the KWIC-concordance to match some secondary search criterion alongside the original keyword.

Say the user has got a KWIC-concordance with some common word as keyword and she is only interested in the lines where the keyword is preceded by some other word. By entering this other word into the *String*-field and pressing the *Force*-button, the user will tell *Lemmie* to pick those KWIC-lines from the concordance that contain this other word as well as the keyword and hence narrow the concordance.

The search window is important here, as it specifies where the forcing collocate must occur to be valid, for example at a maximum distance of two tokens to the left of the keyword. The default window is three tokens at both sides of the node, but it can naturally be changed.

Note that *Lemmie* will not take into consideration any changes made to the size of the search window after the original KWIC-lines have been fetched. If you want to change the window before forcing a collocate, you must re-retrieve the KWIC-lines first with the correct window size set. This might seem

confusing; however, it makes forcing a collocate much more efficient, as *Lemmie* does not have to fetch words in the window more than once.

It is possible to force collocates again and again, hence narrowing the concordance further. Obviously, the forcing collocate can be any valid search statement, i.e. any statement that could be used during lexical lookup in the database (a wordform or a part thereof, a baseform, a POS-tag, a MSD-tag or any combination of these)³.

2.5 Getting more context

If the KWIC-line does not contain enough context to meet the user's needs, he can see the article/page that a specific KWIC-line is fetched from by double-clicking the line in question. This will open the text in another window.

This window has a pull down-menu, from which one can choose how the wordforms in the article are to be presented. The default is "as is", i.e. as running tokens. However, the user can choose to see the text as running lemmas (baseforms), running tokens and lemmas, running tokens and part-of-speech tags and so on.

2.6 Fetching collocations

In addition to letting the user specify forcing collocates, *Lemmie* can also be asked to calculate the significant collocations to some keyword by itself.

Calculating collocations is done in the same fashion as the fetching of KWIC-lines. The user chooses the keyword to be used, either a word form, a baseform (which will make all available inflections keywords as well) or a morphosyntactic description.

When a keyword has been selected, the user asks *Lemmie* to calculate the collocations by pressing the *Get Collocations*-button. The specified search window will be taken into account in the calculus. The formula to be used is set in the *Statistics*-menu.

If the checkbox labeled *Base form query* is ticked, *Lemmie* will build a collocation table of baseforms (ergo all inflections/derivations are considered as instances of the baseform). Otherwise the collocation table will contain inflected forms as well as possible baseforms.

In the first version of *Lemmie* five different statistics for calculating significant collocations are implemented: the Dice coefficient, Ted Dunning's loglikelihood ratio, Mutual Information, Specific Mutual Information and T-score. (See for example Dunning 1993, pp. 71—74, Smadja et al. 1996, pp. 8—9, Brown et al 1992, p. 477 and Church & Mercer 1993, pp. 19—20).

³ When using a forcing collocate full regular expressions are permitted in the *String*-field. (This is not the case during lexical lookup.)

2.7 Fetching colligations

A colligation is a collocation of grammatical and surface forms or grammatical and grammatical forms. An example is the combination of a verb and the wordform "karkuun" (as in "päästä karkuun", "juosta karkuun", "lähteä karkuun", "yrittää karkuun", all meaning something like "make a break", "run away").

Lemmie will calculate colligations to a selected node when the user presses the *Get Colligations*-button in the main window. Possible nodes are a wordform, a baseform or a morphosyntactic description (which automatically includes the part-of-speech tag). Ticking the *Base form query*-checkbox will ask *Lemmie* to calculate only the POS-tags that collocate with the keyword. Otherwise full morphosyntactic descriptions are used.

The same options apply to the colligational as to the collocational calculus, i.e. the size of the search window and the statistic to be used can be specified by the user.

2.8 An example

So far, all aspects of *Lemmie* have been described but not demonstrated. Thus, to clarify what has been said, an example is surely in order. Here is an example of a typical search, including both datadriven and databased features and starting with a lexical lookup.

The following screenshot shows the result of this example:

Screenshot 2. An example of the use of *Lemmie*.

The screenshot shows the Lemmie (version 1.0) interface. The search string is 'puhaltaa'. The results are displayed in a text area and a table below.

Word forms	Freq	Files	Base forms	Freq	Inflects	POS	Morphosyntactic description
hiili	10	10	hiili	95	8	Noun	Nom SG

At the bottom of the interface, a status bar indicates: 'Lemmie narrowed the results with 'puhaltaa' [All] [.*?.*?]. 29 line(s) matched. (20 showing.)'

Say we have a user who wants to find out what multi-word patterns or phrasal entities there are where the word "hiili" ("coal", Noun) plays some important role.

First, "hiili" has to be fetched from the database. The user enters "hiili" as the search string and presses *Lookup*.

If *Keskisuomalainen* 1999 (another of the available newspapers in SKTP in September 2000) is the currently active subcorpus, the user will find out that the wordform "hiili" is found ten times in ten different articles. Further, she discovers that the baseform is "hiili" and that the sum of all frequency counts of all the available inflections of this word in *Keskisuomalainen* 1999 is 95. In addition, there are eight different inflections present. The word is a noun and the form searched for is the same as the baseform, i.e. nominative singular.

Next, the user wants *Lemmie* to find those collocates that might be of some importance around "hiili" in any inflection. Therefore she selects the baseform from the second database query box and ticks the *Base form query*-checkbox. The search window is set to three tokens on any side of the node word by default, and this is fine for now. Similarly the default statistic, Ted Dunning's loglikelihood ratio, is used.

Finally, the user presses the *Get Collocations*-button and *Lemmie* starts counting collocational information around the baseform "hiili". After a few seconds the collocates are shown in another window (screenshot 3).

Screenshot 3. Lemmas collocating with "hiili" in any inflection.

Value	Count	Collocate
42.67055	32	hiili + yksi
35.59586	32	hiili + ja
35.40199	29	hiili + puhaltaa
10.71236	21	hiili + olla
2.06312	7	hiili + puhaltaminen
1.50198	7	hiili + ei
1.49798	6	hiili + yhteinen
1.05244	5	hiili + tulinen
1.04971	5	hiili + öljy
1.03435	5	hiili + lisätä
0.87007	5	hiili + saada
0.67278	4	hiili + rauta
0.67130	4	hiili + polttaa
0.65347	4	hiili + käyttö
0.64259	4	hiili + käyttää
0.64100	4	hiili + sama
0.60926	4	hiili + tai
0.59582	4	hiili + kuin
0.56165	4	hiili + voida
0.43629	4	hiili + että

In this collocational table it becomes quite clear that "yksi" ("one", Numeral), "yhteinen" ("collective", "shared", Adjective), "puhaltaa" ("to blow", Verb) and "puhaltaminen" (nominal form of "to blow", i.e. something like "the blowing") play some major role around "hiili".

What might this role be? To find out, the user asks *Lemmie* to fetch the KWIC-lines around "hiili" in any inflection by pressing the *Get Concordance*-button with the baseform "hiili" still selected (thus highlighted) in the query box. *Lemmie* fetches all the 95 KWIC-lines for him, so there is still one step to take, until the collocations are clearly revealed in the concordance.

What the user needs to do is use a forcing collocate. So, he enters the string "puhaltaa" in the *String*-field, makes sure the *Base form query*-checkbox is still ticked and presses the *Force*-button. Almost instantly, *Lemmie* has narrowed the concordance to those KWIC-lines that contain the baseform "puhaltaa" in some inflection within a window of three words at any side of the node.

A pattern becomes clear; it seems, "puhaltaa yhteen hiileen" (literally translated as "blowing into one coal", which means something like "pulling together" or "working towards some shared goal") is a Finnish metaphor and it can be found 29 times in the *Keskisuomalainen* subcorpus of 1999.

To conclude, this example shows how one can use a semi-datadriven approach to find phrasal patterns in the Bank of Finnish through *Lemmie* and the lexical database SKTP-LT. Naturally, there are many other types of linguistic questions that can be at least partially answered with the help of *Lemmie*. Certainly, there are still a lot of possible enhancements one would want to see implemented in the database and *Lemmie* in the future. In the next section we will have a look at some of them.

3 Some possible expansions of Lemmie and the lexical database

As we saw earlier, SKTP-LT contains quite a lot of useful statistic and grammatical information about the syntactically disambiguated types that occur in the Bank of Finnish. This information can be fetched through *Lemmie*, the corpus query tool developed for SKTP.

However, to be really useful in, for instance, lexicographic work, the types and particularly the baseforms in the database should be semantically disambiguated as well.

At the moment the homographic wordforms "erottaa" (Verb Nom SG Act linf, "to keep apart/separate/divide", the baseform) and "erottaa" (Verb Pr Act Ind S 3P, "[he/she/it] keeps apart/separates/divides") are being kept apart, thanks to the syntactic disambiguation of the corpus.

Nevertheless, the baseform "erottaa" has four major senses or lexemes, according to the dictionary *Suomen kielen perussanakirja* (1990, p. 124, entry "erottaa"). Therefore, one would like these lexemes to be marked in the database as well, so that every occurrence of a wordform that had "erottaa" as baseform was tagged with the lexeme number in question. In this way, one could separate the wordform into at least eight different database entries, two for each lexeme, i.e. the baseform and the present tense.

Moreover, one could quite easily connect the senses to their definitions by extracting these from some regular dictionary and storing them in the database. Clearly, one would have to leave unknown words without a definition at this point.

The above might be a little far fetched, since there are no reliable, fully automatic semantic taggers available for Finnish. Therefore, one solution could be to allow the user to edit the corpus files and the database himself. Not directly, of course, but in some other fashion. The user could, for example, edit a personal copy of some fraction of SKTP, possibly adding semantic information or such. The changes would then be stored in some databaselike files in the user's home directory and be retrievable later. In this way, the changes made by the user could at any time be run concurrently with the static database files, i.e. the user could tailor *Lemmie* to fit his own specific needs.

One could take the above argument even further and add an option to *Lemmie*, so that the user could use text materials of his own through the program. These should naturally be formatted according to TEI-standards, so that *Lemmie* could use the files properly.

Except for editability, there are some other features that could be added to SKTP-LT and *Lemmie*. For instance, the most significant collocates for each and every type could be stored in the database, as well as the most common phrases and multi-word patterns. These could be fetched more or less automatically from the corpus using some statistic, e.g. Ted Dunning's loglikelihood ratio.

Certainly, collocates, phrases and patterns can be calculated on the fly for a particular word or baseform using *Lemmie* or some other tool, so the advantage of running the whole corpus through this process can be questioned. However, the ideas on possible expansions of the database and *Lemmie* that have been

presented here hopefully serve as an example of how an already quite broad corpus service can be widened further.

Acknowledgements: I am grateful to Pia Virtanen and especially Kari Pitkänen for their comments on the manuscript. Further, my sincerest thanks go to Kimmo Koskenniemi for letting me develop *Lemmie* and the lexical database alongside my actual duties at SKTP and to Antti Arppe for his ideas on the possible enhancements of the system. Sari Salmisuo has also been of great assistance sharing the burden of tagging and validating the corpus texts. I am also thankful to Manne Miettinen and the rest of the crew involved at CSC Scientific Computing. Without their encouragement *Lemmie* and SKTP-LT would never have evolved from the prototype stage.

References

- Brown, P. F. et al.** 1992. "Class-Based n-gram Models of Natural Language". *Computational Linguistics*, Volume 18, Number 4, pp. 467—480.
- Butler, C. S.** 1998. "Collocational Frameworks in Spanish". *International Journal of Corpus Linguistics*, Volume 3, Number 1, pp. 1—32.
- Church, K. W. & Mercer, R. L.** 1993. "Introduction to the Special Issue on Computational Linguistics Using Large Corpora". *Computational Linguistics*, Volume 19, Number 1, pp. 1—24.
- Dunning, T.** 1993. "Accurate Methods for the Statistics of Surprise and Coincidence". *Computational Linguistics*, Volume 19, Number 1, pp. 61—74.
- Firth, J. R.** 1957. *Papers in Linguistics 1934—1951*. London: Oxford University Press.
- Karlsson, F.** 1986. "Frequency Considerations in Morphology". *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung* 39, pp. 19—28.
- Smadja, F.** 1993. "Retrieving Collocations from Text: Xtract". *Computational Linguistics*, Volume 19, Number 1, pp. 143—177.
- Smadja, F & et al.** 1996. "Translating Collocations for Bilingual Lexicons: A Statistical Approach". *Computational Linguistics*, Volume 22, Number 1, pp. 1—38.
- Sperberg-McQueen, C. M. & Burnard, L.** 1994. *Guidelines for Electronic Text Encoding and Interchange (TEI P3)*, Vol. 1 & 2. Chicago: ACH/ACL/ALLC Text Encoding Initiative. (Also available on the Internet: <http://etext.virginia.edu/TEI.html>)
- Suomen kielen perussanakirja* 1990. Volume 1, A–K. Kotimaisten kielten tutkimuskeskuksen julkaisu 55. Helsinki: VAPK-kustannus.

Appendix A — Getting started with Lemmie

If you want to use Lemmie and other services at the Bank of Finnish, you will need a computer user account at the corpus server (`corpus.csc.fi`) at the Center for Scientific Computing in Espoo, Finland. To obtain this account, please send mail to the following address asking for the proper forms (or download them at <http://www.csc.fi/kielipankki/sktp/>):

CSC — Center for Scientific Computing
Kielipankki – luvat
PL 405
02101 ESPOO

Once you have signed the user agreement and received an account at the corpus mainframe you can start using *Lemmie* and the lexical database SKTP-LT.

Log on to the corpus server and type the following:

```
corpus.csc.fi> lemmie &
```

If your path is set up correctly⁴, *Lemmie* should respond with the following message in the terminal:

```
Starting Lemmie. Hold on, please.  
Connected to the DB.
```

After this *Lemmie*'s main window (see screenshot 1) will appear on screen and you can start using the tool. Help is available in the *Lemmie* manual, accessible through the File-menu.

If *Lemmie* does not start for some reason, please do not hesitate to contact the corpus server administrator at CSC, specifying what exact error message you received.

However, some errors are easily fixed, so here is a list of common errors and their solutions:

```
Fatal I/O error
```

This is typically a Windows problem. On a Windows (NT or 98) system you will need to have an X-server installed, to be able to run *Lemmie*. Otherwise, you will constantly receive this I/O error message and *Lemmie* will not start.

```
Couldn't connect to display ":0"
```

Linux/Unix: Apparently your `$DISPLAY`-variable is not set up correctly. Correct this and try again. *Windows*: Enable the "Forward X11 connection"-option in

⁴ The path to *Lemmie* is `/l/bin/lemmie`, so make sure that `/l/bin` is set in the `$PATH`-variable in your `.bash_profile` file.

your SSH-client and restart your session (log off from the corpus server and log on again)

```
FATAL: Sorry, you don't have permission to run Lemmie
```

When you receive this message, you do not have the right group privileges on the corpus server. This problem you cannot fix yourself. Contact the system administrator.

Appendix B — A brief note on regular expressions in Lemmie

These notes are primarily useful when building a regular expression to be used in the *MSD*-field. In the *String*-field the use of regular expressions is restricted during lexical lookup in the database, due to speed limitations. When forcing a collocate regular expressions are also permitted in the *String*-field.

- . Matches one arbitrary character

Example: `a.` matches "aa", "ab", "ac" etc.

- * Matches zero or more of *the previous character*. (There is a discrepancy here: in the *String*-field in *Lemmie*, the star matches zero or more of *any* character. This is due to the fact that the *String*-field does not accept full regular expressions during lexical lookup. To imitate this behaviour in the *MSD*-field, use the `.*` notation.)

Example (MSD-field): `a*` matches "a", "aaa", but not "abc", `a.*` matches "a", "aaa" and "abc" but not "bcd"

Example (String-field): `a*` matches "a", "aaa", "abc"

- + Matches one or more of the previous character

Example: `ab+c` matches "abc", "abbc", "abbbc", but not "ac"

- ? Indicates that the previous character may or may not occur

Example: `ab?c` matches "ac" and "abc"

- ^ Matches the beginning of input⁵

Example: `^word.*` matches "wordform", "word formation" and "word sense disambiguation", but not "multi-word" or "keyword"

- \$ Matches the end of input

Example: `.*word$` matches "keyword", "stopword", "unknown word", but not "word formation" or "words"

- (`x|y`) One of the strings *x* or *y* in brackets matches

⁵ Note that input is normally either a wordform or a morphosyntactic description in Lemmie, not a concordance line or similar.

Example: `perusta(mme|tte)` matches "perustamme" and "perustatte"

[ab] One of the individual characters inside the square brackets match

Example: `[abc]` matches "a", "b" or "c", but not "d" or "e"

[^ab] One character *not* inside the square brackets match

Example: `[^abc]` matches "d", "e" or "f", but not "a", "b" or "c"

{m,n} Two digits separated by a comma can be placed in curly brackets, {n,m}, to indicate that the previous word *must* occur at least *n* times and *can* occur at most *m* times in a row. If *n* is left out, {,m}, *n* is interpreted as zero. If *m* is left out, {n,}, *m* is interpreted as infinite.

Example: `ab{1,2}a` matches "aba" and "abba", but not "aa" or "abbba"

For a more extensive tutorial on the use of regular expressions, see

Friedl, J. E. F. 1997. *Mastering Regular Expressions*. Sebastopol: O'Reilly & Associates, Inc.

Appendix C — Abbreviations

<i>GUI</i>	Graphical User Interface
<i>KWIC</i>	KeyWord In Context
<i>MSD</i>	MorphoSyntactic Description, e.g. "Nom SG"
<i>MySQL</i>	The database system that the lexical database is stored in, see http://www.mysql.org/
<i>POS</i>	Part-Of-Speech, e.g. "Verb"
<i>SGML</i>	Standard Generalized Markup Language
<i>SKTP</i>	Suomen kielen tekstipankki (the Bank of Finnish)
<i>SKTP-LT</i>	Suomen kielen tekstipankin leksikaalinen tietokanta (the Lexical Database of the Bank of Finnish)
<i>SQL</i>	Structured Query Language, a standardized query language for requesting data from a database
<i>TEI</i>	Text Encoding Initiative, see http://etext.virginia.edu/TEI.html

Appendix D — The Finnish tag set in SKTP

The Finnish part of the Bank of Finnish has been tagged with *Textmorfo*, an automatic morphosyntactic tagger provided by Kielikone (see <http://www.kielikone.fi/> for additional information).

This non-comprehensive list of tags is intended to be a guide for the *Lemmie* user only. It is *not* a complete or fully correct paradigm of the morphosyntactic distribution in Finnish, nor a thorough description of the *Textmorfo* tag set as a whole.

Elements in the morphosyntactic description

POS *Abbrev, Adjective, Adverb, Code, CompPart, Conjunction, Interjection, Noun, Numeral, Preposition, Pronoun, Proper, Verb*

Case *Ab* (Abessive), *Abl* (Ablative), *Acc* (Accusative), *Ad* (Adessive), *All* (Allative), *Com* (Comitative), *El* (Elative), *Ess* (Essive), *Gen* (Genitive), *Genom* (nominalized word in Genitive), *Ill* (Illative), *In* (Inessive), *Instr* (Instructive), *Nom* (Nominative), *Part* (Partitive), *Transl* (Translative)

Grade *Comp* (Comparative), *Sup* (Superlative)

Modal *Iinf* (1st Infinitive), *Iiinf* (2nd Infinitive), *IIIinf* (3rd Infinitive), *IVinf* (4th Infinitive), *Vinf* (5th Infinitive), *Ipartic* (Present Participle), *Iipartic* (Perfect Participle), *Ind* (Indicative), *Cond* (Conditional mood), *Imper* (Imperative), *Pot* (Potential mood)

Num *SG* (Singular), *PL* (Plural)

Person *1P* (1st person), *2P* (2nd person), *3P* (3rd person)

PossSuff *S* (Singular possessive suffix), *P* (Plural possessive suffix)

Tense *Pr* (Present tense) *Imp* (Imperfect)

Voice *Act* (Active), *Pass* (Passive)

Some useful morphosyntactic patterns

Elements surrounded by curly brackets are optional. (Moreover, some combinations are not possible in Finnish.)

Abbreviation:	Case Num
Adjective:	{Case} {Num} {{PossSuff} Person} {Grade}
Adverb:	{Case} {Num} {{PossSuff} Person} {Case} {Num} {Grade}
Noun:	{Case} {Num} {{PossSuff} Person} {Case} {Num} {Grade}
Numeral:	{Case} {Num} {{PossSuff} Person}
Preposition:	{Case} {Num} {{PossSuff} Person} {Grade}
Proper:	{Case} {Num} {{PossSuff} Person}
Pronoun:	{Case} {Num} {{PossSuff} Person} {Case} {Num} {Grade}
Verb:	{Case} {Num} {Voice} {Modal} {{PossSuff} Person} {Grade} {Tense} {Voice} {Modal} {{PossSuff} Person}

Some examples

Abbreviation	<i>Nom SG</i> <i>Gen SG</i>	"AIESEC", "DOS", "SEA" "OKO:n"
Adjective	<i>Abl PL</i> <i>Part SG P 1P Sup</i> <i>Transl PL Sup</i>	"akateemisilta", "edestavastuuttomilta" "parastamme" "ankarimmiksi", "imuvoimaisimmiksi"
Adverb	<i>Gen SG</i> <i>Sup</i>	"äärimmäisen" "köpelöimmin", "silmiinpistävimmin"
Noun	<i>Com PL 3P</i> <i>Instr PL</i> <i>Transl SG P 1P</i> <i>Ess SG Comp</i>	"aikapommeineen", "takiloineen" "faksitiedottein", "nettisivuin" "hauskutukseksemme", "jäseniksemme" "illempana", "täyteläisempänä"
Numeral	<i>Ad PL</i> <i>In SG 3P</i>	"sadoillatuhansilla", "yksillä" "ensimmäisessään"
Preposition	<i>Ill SG 3P</i> <i>S 1P</i>	"alle", "ulottuvilla" "haltuunsa" "eteeni", "hallussani"

Proper	<i>Gen PL</i> <i>Ad SG 3P</i>	"Ahtisaarten", "Rabatti-ketjujen" "Benettonillaan", "Finnjetilläään"
Pronoun	<i>Abl PL</i> <i>Acc PL P 1P</i> <i>El SG Comp</i>	"harvoilta", "muilta" "meidät" "useammasta"
Verb	<i>Com PL Act Ipartic</i> <i>El SG Pass IIpartic Sup</i> <i>Imp Act Ind P 1P</i> <i>Pr Act Pot S 2P</i>	"aaltoilevine", "kapinoivine" "ihailluimmasta" "asensimme", "ylitimme" "edennet", "tehnet"

Appendix E — The texts in SKTP

This table describes the texts in SKTP in September 2000.

Material	Tokens	MSD	Texttype	Lang
Demari (1998-2000)	5 270 000	Tagged	Newspaper	FI
Finska notisbyrån (1999-2000)	9 780 000	Tagged	News agency	SE
Hämeen Sanomat	1 240 000	Tagged	Newspaper	FI
Helsingin Sanomat 1995	21 560 000	Untagged	Newspaper	FI
Helsingin Sanomat 1996	21 240 000	Untagged	Newspaper	FI
Helsingin Sanomat 1997	8 280 000	Untagged	Newspaper	FI
Hufvudstadsbladet 1998	9 300 000	Tagged	Newspaper	SE
Hufvudstadsbladet 1999	9 720 000	Tagged	Newspaper	SE
Hyvinkään Sanomat (1994 & 1997)	2 010 000	Tagged	Newspaper	FI
Jakobstads Tidning (1999-2000)	3 330 000	Tagged	Newspaper	SE
Kaleva	9 760 000	Tagged	Newspaper	FI
Kangasalan Sanomat	270 000	Tagged	Newspaper	FI
Karjalainen 1991	2 650 000	Tagged	Newspaper	FI
Karjalainen 1992	4 440 000	Tagged	Newspaper	FI
Karjalainen 1993	3 140 000	Tagged	Newspaper	FI
Karjalainen 1994	4 890 000	Tagged	Newspaper	FI
Karjalainen 1995	6 000 000	Tagged	Newspaper	FI
Karjalainen 1996	6 190 000	Untagged	Newspaper	FI
Karjalainen 1997	6 280 000	Tagged	Newspaper	FI
Karjalainen 1998	5 970 000	Tagged	Newspaper	FI
Karjalainen 1999	2 160 000	Tagged	Newspaper	FI
Keskisuomalainen 1999	7 930 000	Tagged	Newspaper	FI
Suomen Kuvalehti 1995-1999	1 000 000	Untagged	Periodical	FI
Turun Sanomat 1998	10 740 000	Tagged	Newspaper	FI
Turun Sanomat 1999	11 110 000	Tagged	Newspaper	FI
WSOY/Children's books	130 000	Untagged	Books	FI
WSOY/Fiction	700 000	Untagged	Books	FI
WSOY/Non-fiction	510 000	Untagged	Books	FI
Other books	170 000	Untagged	Books	FI

Technical Reports
from the Department of General Linguistics,
University of Helsinki

- TR-1 **Timo Järvinen & Pasi Tapanainen** 1997. *A Dependency Parser for English*, 43 pp.
- TR-2 **Pirkko Suihkonen** 1998: *Documentation of the Computer Corpora of Uralic Languages at the University of Helsinki*, 74 pp.
- TR-3 **Mickel Grönroos** 2000: *The Lexical Database of the Bank of Finnish*, 27 pp.

ISBN 951-45-9563-7
ISSN 1238-6995
Helsinki, September 2000
University of Helsinki