Authors:	Kettunen Kimmo, Kunttu Tuomas, Järvelin Kalervo
Name of article:	To stem or lemmatize a highly inflectional language in a probabilistic IR environment?
Year of publication:	2005
Name of journal:	Journal of Documentation
Volume:	61
Number of issue:	4
Pages:	476-496
ISSN:	0022-0418
Discipline:	Natural sciences / Computer and information sciences
Language:	en
School/Other Unit:	School of Information Sciences

URN: http://urn.fi/urn:nbn:uta-3-744
DOI: http://dx.doi.org/10.1108/00220410510607480

To stem or lemmatize a highly inflectional language in a probabilistic IR environment?

Kettunen, Kimmo, Kunttu, Tuomas and Järvelin, Kalervo

University of Tampere
Department of Information Studies, Kanslerinrinne 1, FIN-33014, University of Tampere, Finland
lokike@uta.fi

University of Tampere
Department of Information Studies, Kanslerinrinne 1, FIN-33014, University of Tampere, Finland
Tuomas.Kunttu@uta.fi

University of Tampere
Department of Information Studies, Kanslerinrinne 1, FIN-33014, University of Tampere, Finland
likaja@uta.fi

Keywords: morphological processing, probabilistic IR, comparison of methods

Effects of three different morphological methods - lemmatization, stemming and inflectional stem generation - for Finnish are compared in a probabilistic IR environment (INQUERY). Evaluation is done using a four point relevance scale which is partitioned differently in different test settings. Results show that inflectional stem generation which has not been used much in IR, compares well with lemmatization in a best-match IR environment. Differences in performance between inflectional stem generation and lemmatization are small and they are not statistically significant in most of the tested settings. It is also shown that hitherto a rather neglected method of morphological processing for Finnish, stemming, performs reasonably well although the stemmer used – a Porter stemmer implementation – is far from optimal for a morphologically complex language like Finnish. In another series of tests, the effects of

compound splitting and derivational expansion of queries are tested.

Introduction

Morphological variation of the search keys and its handling for IR purposes has been studied quite extensively during the last decades. Four different language dependent methods exist for the handling of morphological variation: lemmatization, stemming, stem generation and full form generation of words. Also a variety of language independent methods, such as n-grams and other fuzzy matching methods have been proposed. Some of the language dependent methods can be considered better for a certain morphological type of language. Some do not fit a specific language at all. Lemmatization has been successful for a variety of languages, but also stemming has still been used extensively. New insights have been gained and more interesting languages have also been studied during the 1990's with stemming.

Several morphological methods have been used in best-match environments. *Stemming,* i.e. many-to-one mapping where semantically related distinct word forms are reduced to identical stems, has been quite popular with several languages. Popovič and Willet (1992) showed that both stemming and manual truncation work well for Slovene in a best-match environment (INSTRUCT). Mayfield and McNamee (2003) have tested stemming and different *n-gram* methods for a variety of languages, which include e.g. Swedish, German, and Finnish. Sever and Bitirim (2003) tested three different types of stemmers for Turkish in a vector space model (SMART) and found stemming a suitable method for Turkish. It increased search precision by approximately 25 % when compared to no stemming at all. Tomlinson (2002, 2003) describes results for 8 − 9 European languages using lexical and algorithmic stemming. In Kraaij and Pohlman (1996) and Braschler and Ripplinger (2003) several different types of stemmers are introduced for Dutch and German, respectively. Other morphologically interesting languages studied recently include for example Amharic (Alemayehu and Willet, 2003), Arabic (e.g. Abu-Salem and others, 1999), Latin (Schinke and others, 1996), and Portuguese (Silva and Oliveira, 2003). The morphological complexity of the languages in these studies varies, but all of the studies include at least one language that is morphologically somewhat complex. However, it is far from obvious which morphological method one should use in a best-match environment for highly inflectional languages.

Finnish is well known for its morphological complexity. Due to the rich morphology of Finnish inflected word forms or ad hoc truncated general forms are not considered as good search keys for information retrieval in Finnish documents. Some kind of morphological processing of the search keys is needed for getting satisfactory results. In the case of Finnish, however, e.g. stemming has not been considered a suitable method and full form generation is clearly unpractical due to the number of different possible surface forms. The computational burden of 28 (2*14) case forms in singular and plural would already be high for the retrieval system. Moreover, Finnish nouns may in principle have about 2200 inflected forms – many of them quite unlikely, though (Karlsson, 1983, 1986). Therefore the Finnish language is a particularly suitable environment for comparing alternative methods for morphological processing in a best-match environment.

Automatic stem generation and lemmatization programs for Finnish have been implemented since early 1980's and they have been used in information retrieval systems. The most prominent stem generation programs have been Finstems (Koskenniemi, 1985) and Hahmotin by Kielikone Ltd. (Alkula, 2000). Given an input word in base form, these programs are able to generate all the varying inflectional stems for the word. Depending on the input noun, 1 – 5 different stems (including the base form) are produced for a noun. These stems cover all the variation that occurs in the stems, and the grammatically possible ca. 2200 inflected forms are produced by concatenating case endings, possessive endings, clitics etc. after each other to appropriate stems. Thus the approach of using inflectional stems in IR to cover morphological variation of the search keys is quite promising. All stemmers, in particular stemmers based on suffix stripping, do not take this feature of Finnish into account.

There are also at least three different morphological lemmatizers for Finnish: FINTWOL (Koskenniemi, 1983), Morfo (Jäppinen and Ylilammi, 1986) and Ment (Blåberg, 1994). A lemmatizer analyzes inflected word forms and returns their base forms. If an inflected word form is ambiguous, several base forms are returned. Analysis of a lemmatizer is based on a set of rules and use of a large lexicon with tens of thousands of entries.

General overall evaluation of the value of lemmatization and stem generation programs for Finnish IR in a Boolean retrieval environment has been published recently in (Alkula, 2000, 2001). The programs evaluated were Hahmotin, FINTWOL and Morfo. Morphological lemmatization and stem generation were tested with different types of indexes (lemmatized index, with compounds split and compounds as such, and inflected index) and manual truncation of the search keys by a seasoned user was tried out as well. Two of these methods, lemmatization and stem generation, are studied in this study in a best-match system. Two other methods, namely stemmed query keys and plain unprocessed query words as such are also tested.

In this paper, we show the performance results of lemmatization, stem generation and stemming for Finnish. The research problems of this paper are as follows:

- How do lemmatization and inflectional stem generation compare in a probabilistic environment?
- Is a stemmer a realistic alternative for handling of the morphology of a highly inflectional language, such as Finnish, for IR?

- Is simulation of truncation feasible in a best-match system?

As sub-problems compound splitting, derivational queries, and different types of topics are studied.

In the next section we present data and methods of the study; after that stemming, stem generation and lemmatization are discussed and defined**.** Following that we present our results. The two last sections contain discussion and conclusions.

Data and methods

The tests of this study were conducted in the Information Retrieval Laboratory of the Department of Information Studies, University of Tampere. Actual searches were conducted with a probabilistic partial match system, INQUERY, version 3.1 (Callan and others, 1992, Broglio and others, 1995) with two partly different testing environments.

The test collection, TUTK, contains a full text database of newspaper articles published in three Finnish newspapers in 1988 − 1992. The newspapers are Aamulehti (a general newspaper), Keskisuomalainen (a general newspaper) and Kauppalehti (an economics oriented five day newspaper) and the database consists of 53 893 articles. The articles represent different sections of the newspapers, mostly economics (from all sections of Kauppalehti, some 16 000 articles), and foreign and international affairs (Aamulehti, some 25 000 articles) and articles from all sections of Keskisuomalainen (some 13 000 articles). (Sormunen, 2000, Kekäläinen, 1999).

Articles of the database are fairly short on average. Typical text paragraphs are two or three sentences in length. The topic set consists of 30 topics (Sormunen, 2000). Topics are long: the mean length of the original topics is 17.4 words. When stop words are omitted, the mean length is 15.06 words per topic. Statistics derived from the text database and the associated topics are shown in Table I.

Table I. Statistics of the text database and topics

Characteristics of TUTK and topics			**Source of information**
1.	Number of documents	53 893	Sormunen (2000), INQUERY count
2.	Number of topics	30	Sormunen (2000)
3.	Mean words per topic, original	17.4	Computed
4.	Mean words per topic, stop words removed	15.04	Computed
5.	Mean number of words per document	218.06	computed (#7/#1)

6.	Number of inflected word-form types in the database	709 317	size of the inflected index, computed
7.	Number of word-form tokens in the database	11 752 290	word-form types * frequencies
8.	Mean length of word-form types in characters	13.14	Computed
9.	Mean length of the word-form tokens weighted with the frequency of the words	8.03	Computed

Relevance levels of TUTK

Sormunen (2000, p. 63) describes the relevance levels of TUTK test collection. A four point scale is used and its interpretation is in Table II.

Table II. Relevance levels of TUTK interpreted.

Relevance	**Document is**

level	
0	totally off target.
1	marginally relevant, refers to the topic but does not convey more information than the topic description itself.
2	relevant, contains some new facts about the topic.
3	highly relevant, contains valuable information, the article’s main focus is on the topic.

In the first testing environment of this study, *Environment One*, queries were both performed on separate relevance level 3 and on binary relevance level. Binary relevance level was created from the original four levels by combining levels 2 and 3 as relevant; levels 0 and 1 were considered irrelevant. Relevance level 3 is referred to in performance tables as *stringent*, and binary level as *normal*.

In *Environment Two* (Kunttu, 2003), the relevance scales of the TUTK collection were used in a slightly different way. Here *liberal* scale included all the relevance levels 1 – 3, *normal* and *stringent* scales were the same as in Environment One.

Methods

The main purpose of this study was to compare three different morphological methods in a best-match IR environment. Three of the programs, Finstems, FINTWOL1[1], and Snowball[2], were obtained from external sources. The fourth program, MaxStemma, was implemented by the first author in early 1990's. Its original version is described in more detail in (Kettunen, 1991a, 1991b).

Before going on to the test procedure, a short discussion about terms **lemmatization, stem generation** and **stemming** is necessary.

In linguistics stem is defined as a basic unit from which inflected word forms are generated by adding affixes (Matthews, 1991). This is also the starting point of the approach taken in the development of MaxStemma and Finstems: stems are generated from input base forms for further use. Stemming, then, as it is used in the IR literature has different goals: a stemmer analyses inflected word forms and produces conflated stems that can be used in information retrieval. While linguistic stems are linguistically motivated basic elements of surface form production, stems in IR may be almost anything: stems or roots or ad hoc truncated forms that can be adjusted to varying IR needs.

Stem generators used in this study, MaxStemma and Finstems, work in the following fashion: given the base word form (nominative singular for nouns), they produce all the differing inflectional stems of the words. Depending on the input noun, 1 – 5 different stems (including the base form) are produced for a noun. E.g., if the input word is *kissa ('cat'),* the programs would generate the following inflectional stems for the word*: kissa, kissoi, kissoj.*

Snowball, the stemmer used in this study, returns stems out of inflected word forms. Snowball is a Lovins' style stemmer that strips off suffixes from the input word according to a suffix list and set of rules and returns stems for the words (Frakes 1992, Porter 2001). In the case of Finnish, inflectional forms of the base form *kissa* (‘cat’), e.g. *kissat, kissojen, kissoihin*, are returned by Snowball as *kis*- and *kiso*-. That is, different inflected input words give different output stems, singular and plural forms are not conflated to a single stem in this case.

Lemmatization is most often used to describe the process when inflected word form and its (dictionary) base form are related to each other with an algorithm. Stemming can be seen as a simpler variant of lemmatization (e.g., Jacquemin and Tzoukerman, 1999): some stemming programs may be very sophisticated and use full dictionaries (e.g., Krovetz, 2000) while others (or most) are simpler and do not usually use dictionaries. (Hull, 1996, Harman, 1991, Baeza-Yates and Ribeiro-Neto, 1999, Frakes, 1992, Koskenniemi, 1983, 1985, Paice, 1996).

FINTWOL, the lemmatizer used in this study, analyzes inflected word forms and returns their (dictionary) base forms. Its analysis is based on a set of rules and use of a large lexicon with tens of thousands of entries. In the case of different inflectional forms of *kissa,* a single base form would be returned for all of them.

Stemming thus consists of **analyzing** the inflected word forms at least superficially and returning some base form for them. Stem generation programs are **generative** programs**:** given the base form (nominative singular, the base form used in Finnish dictionaries, in the case of MaxStemma and Finstems), all stem forms of the word are produced to be used as search keys. Inflected word forms are not analyzed by stem generators.

Table III clarifies relationships between stem generation, stemming, morphological lemmatization and generation. Two parameters, the use of a full dictionary and analysis vs. generation are used as the distinguishing features of the systems. Other kinds of features (e.g. rule formulation) could also be used, but these two are considered as the main differentiating features.

Table III. Stemming, stem production and lemmatization

	uses a full dictionary	no dictionary used
analysis	FINTWOL by Koskenniemi (1983) Krovetz' stemmer (2000)	Porter stemmer (1980) Lovins stemmer (1968)
generation	FINTWOL by Koskenniemi (1983)	Finstems by Koskenniemi (1985) MaxStemma by Kettunen (1991a)

Thus **stem generation** will be used in this study to mean production of inflectional stem variants in the manner of Finstems and MaxStemma. **Stemming** is used for Porter and Lovins type of approaches, and **lemmatization** for FINTWOL type of approaches. The demarcation line between stemming and lemmatization, however, is not clear cut. In for example Kraaij and

Pohlman (1996) and Braschler and Ripplinger (2003) several different types of stemmers are introduced for Dutch and German, and it would be fair to characterize some of them as lemmatization programs, since their analysis is based on large dictionaries and is also linguistically motivated. Such a program is also the Krovetz' stemmer (2000), since it uses a full dictionary (The Longman Dictionary of English) to check the proposed stemming before accepting it.[3]

The query process

For the testing of the different methods, two automatic IR laboratory procedures were built. Environment One with stem generation is described schematically in Figure 1.

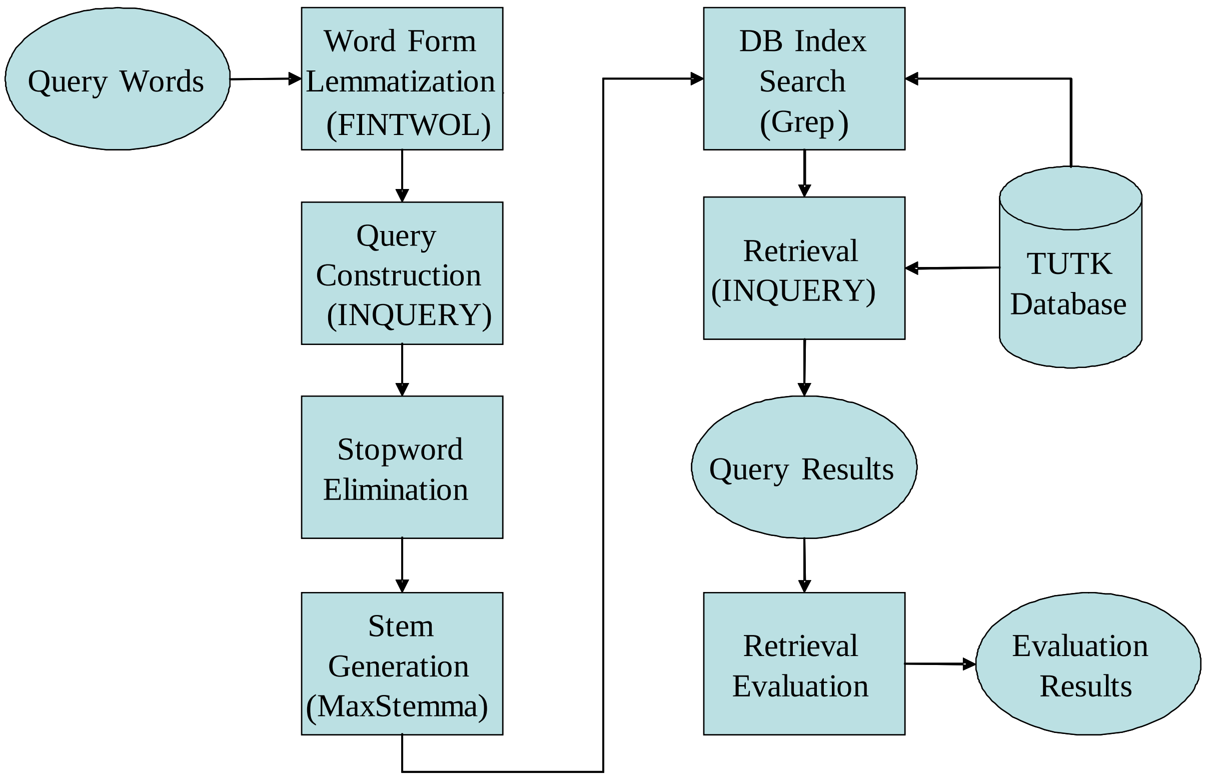


Figure 1. Environment One with stem generation

The basic query process consists of the following steps when *stem generation* is tested in Environment One:

1. Lemmatization of the query words (from topic descriptions) with FINTWOL.

2. Basic INQUERY query generation resulting in a slightly structured query with INQUERY's #sum and #syn operators.

3. Elimination of stop words in the query according to a stop list of Finnish.

4. Generation of stems for final query words with MaxStemma.

5. Grepping (by way of grep command of Unix) of the database index with inflectional stems which will conclude in final query formulation; grepping simulates term truncation, which is not supported in INQUERY.

As an ad hoc example for an input word *kissa* stems *kissa, kissoi, kissoj* would be returned by MaxStemma. By grepping with all these three forms - grep(*"kissa"*|*"kissoi"*|*"kissoj"*) - from the database index, all the inflected forms of the word *kissa* would be found. These matches would be put in the final query.

6. INQUERY retrieval and its evaluation.

When *lemmatization* is used, only steps 1, 2, 3 and 6 are in use. When *stemming* is used, step 1 consists of stemming of the query words. After that, steps 2, 3 and 6 are used. When *plain topic words* are used as query words, only steps 2, 3 and 6 are used.

In Table IV differences of the database indexes for the methods used in Environment One are described briefly.

Table IV. Types of indexes with different methods in Environment One

Method used for query words	**Type of index**
Plain words	inflected index (no processing)
Stemmed words	stemmed index (stemmed with Snowball)
Stem generation	inflected index (no processing)
Lemmatized word forms	lemmatized index (lemmatized with FINTWOL, compounds not split)

As the usage of stem generation was one of the main interests of this study, this procedure is described in more detail here. The process begins with the morphological lemmatization of the topic words; this simulates interactive query, where the user gives the words in their base forms. Morphological lemmatization returns the query words in their base form and compound borders are marked for stem generation. If a compound has more than two parts, only the last compound border will stay marked. Ambiguous or erroneous morphological interpretations are left as such and basic structural query notation of INQUERY is added.

The first topic of the test collection is "**1. George Bushin ja Mihail Gorbatshovin**

tapaaminen Helsingissä syyskuussa 1990. Neuvotteluissa käsitellyt asiat sekä tehdyt päätökset ja sopimukset.” [*Translation: The Bush - Gorbachev Summit in Helsinki in September 1990. The issues of the negotiations, and resolutions and agreements*], and it looks like this after phase 2) (#syn being INQUERY’s synonymy operator, Broglio and others, 1995):

@q1#syn(george) #syn(bush) #syn(ja) #syn(mihail) #syn(gorbatshov) #syn(tapaaminen) #syn(helsinki) #syn(syys/kuu) #syn(1990) #syn(neuvottelu) #syn(käsitellä käsitellä käsitellä) #syn(asia) #syn(sekä) #syn(tehdä tehty) #syn(päätös) #syn(ja) #syn(sopimus)

In this example *#syn(käsitellä käsitellä käsitellä)* is due to the morphological ambiguity of the analysis, which is preserved as such. The other syn clauses contain just one key each at this stage. In *#syn(syys/kuu)*, marking of the compound border for the stem program can be seen.

After the elimination of stop words, the actual procedure of stem generation is started by giving the base word forms to MaxStemma. After the stem generation phase the first query looks like this:

#q 1 = #sum(#syn(george georgei georgee) #syn(bush bushi bushei) #syn() #syn(mihail mihailei mihaili) #syn(gorbatshov gorbatshovi gorbatshovei) #syn(tapaaminen tapaamise tapaamisi tapaamist) #syn(helsinki helsinkei helsinkej helsingi helsingei) #syn(syyskuu syyskui) #syn(1990) #syn(neuvottelu neuvottelui neuvotteluj) #syn(asia asioi asioj) #syn() #syn(tehty tehtyj tehdy tehdyi tehtyi) #syn(päätös päätöksi päätökse) #syn() #syn(sopimus sopimukse sopimuksi));

The empty syn clauses are due to eliminated stop words and do not affect processing.

The draft query is now ready for grepping (cf. Friedl, 1997) of the database index. Grepping of the index simulates term truncation, which is not supported in INQUERY and many other best-match systems. Matching strings are grepped from the index of the database and all the matching words or strings are put into the query. As the result of this phase, the first query will have the following form (only the matches for the first three query words are (partly) listed here, while there are more than 3500 lines of matching words for the whole query). These kinds of formal queries are forwarded to INQUERY, which carries out the queries and evaluates the results.

```
#q1=
#sum(#syn(george georgedalessa georgen georges georgessa georgesta georgete georgetown
georgetownin georgett)

 #syn(bush bush-boris bush-gorba bush-juhla bush-juttu bush-juttuun
bush-jutun bush-kuva bush-nl bush-quayle bush-u bushaastat bushav bushba bushbaltti bushbu
bushbudj bushehrissa bushehrista busheilla busheille busheilta bushhurd bushi bushia bushiakin
...)

#syn(mihail mihailenko mihailenkoa mihailenkolle mihailenkon mihailgorbatshovia mihailiin
mihaililla mihailille mihailin mihailista mihailitshenko mihailitshenkoa mihailitshenkon mihailo
mihailov mihailova mihailovicin mihailovitsh mihailiin mihaililla mihailille mihailin mihailista
mihailitshenko mihailitshenkoa mihailitshenkon)
```

As the index of the database contains all the words of the texts as they occur in them, it is clear that also misspellings will be matched. Misspellings and all other false drops are taken into the final query; no matches from the index are checked in any way during or after this phase in Environment One.

In *Environment two* a morphological lemmatizer (FINTWOL) was tested against a stem generation program, Finstems (Koskenniemi, 1985) with INQUERY. With the morphological lemmatizer the effects of index with split compounds were tested. Along with basic queries, Environment Two used also queries that included the most productive derivatives of the query nouns. This more than doubled the number of the query words in derivational queries. The grepping of the index with stem generation was also done differently in Environment Two. The index was first grepped with the stems, but afterwards the results of the grepping were sifted intellectually or with FINTWOL so that the final query included only the real inflected forms of the query noun.

Shortly listed and explained the methods and index types employed in Environment Two are in table V.

Table V. Methods and index types in Environment Two.

Method / query words	**Index type**
LEMS1, basic query	lemmatized index, compounds split (FINTWOL)
LEMS2, derivational query	lemmatized index, compounds split (FINTWOL)
LEMNS1, basic query	lemmatized index, compounds not split

	(FINTWOL)
LEM2NS, derivational query	lemmatized index, compounds not split (FINTWOL)
INFL1, basic query	inflected index
INFL2, derivational query	inflected index

In Environment Two, queries were analyzed also in a more detailed manner by grouping the queries according to their themes. Four groups were established: purely topical queries, geographically oriented queries, person oriented queries and organizational queries.

Results

In this section, we show the main results of this study. First results in Environment One are shown; next main results in Environment Two are explained.

Test results in Environment One

In the test runs FINTWOL was tested against MaxStemma and Snowball. To get an independent baseline for all the tests, also queries with no morphological processing at all (plain words) were run.

In Table VI average precisions of different methods in Environment One are presented on stringent and normal relevance levels. In Figures 2 and 3 ten point average P-R curves of the same relevance levels are shown.

Table VI. Average precision results in Environment One on the stringent and normal relevance levels, absolute differences between methods in percentage units.

Stringent	**FINTWOL**	**MaxStemma**	**Snowball**	**Plain**
Avg. precision over recall levels (%)	24.1	22.6 (-1,5)	20.0 (-4,1)	12.4 (- 11,7)

Normal	**FINTWOL**	**MaxStemma**	**Snowball**	**Plain**
Avg. precision over recall levels (%)	35.0	34.2 (-0,8)	27.7 (-7,3)	18.9 (-16,1)

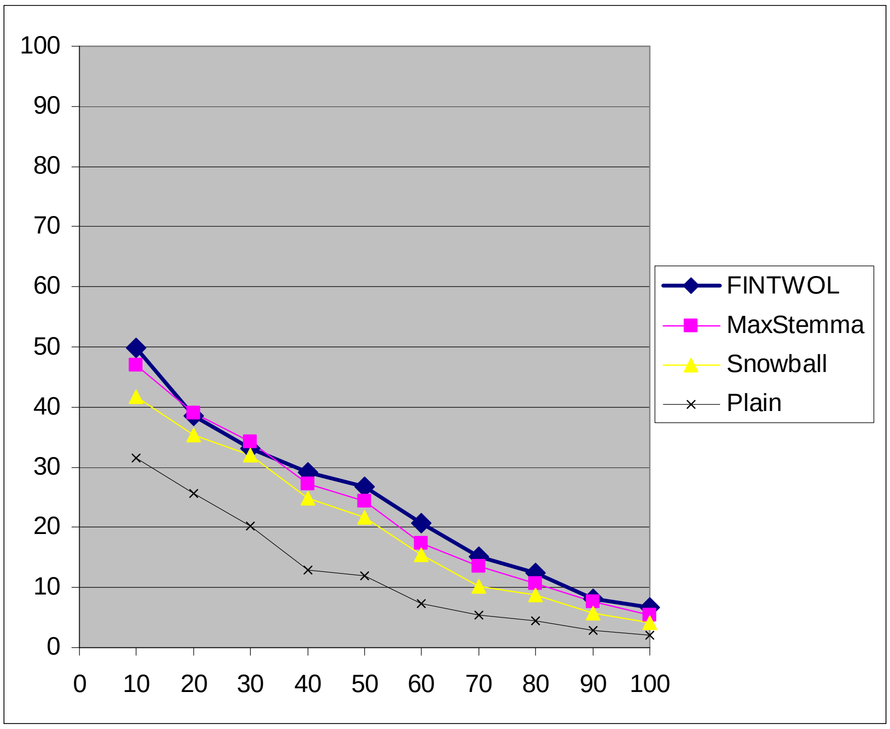


Figure 2. P-R curves on 10 recall levels on stringent relevance level in Environment One.

From Figure 2 one may see that the performance curves of MaxStemma and FINTWOL almost converge from recall level 10 to 40 on the stringent relevance level. After that there is a slightly greater difference until recall level 70, where the curves start to merge again and stay like that until the end. The performance of Snowball is clearly below the former; only at the recall level 30 it performs almost at the same level. It is interesting that performance with Plain Words is ca. 60 − 66 % of the results of FINTWOL until recall level 30; after that the performance of Plain Words falls between ca. 29 – 44 % of FINTWOL's performance. On average Plain Words achieve 51.45 % of FINTWOL's performance level.

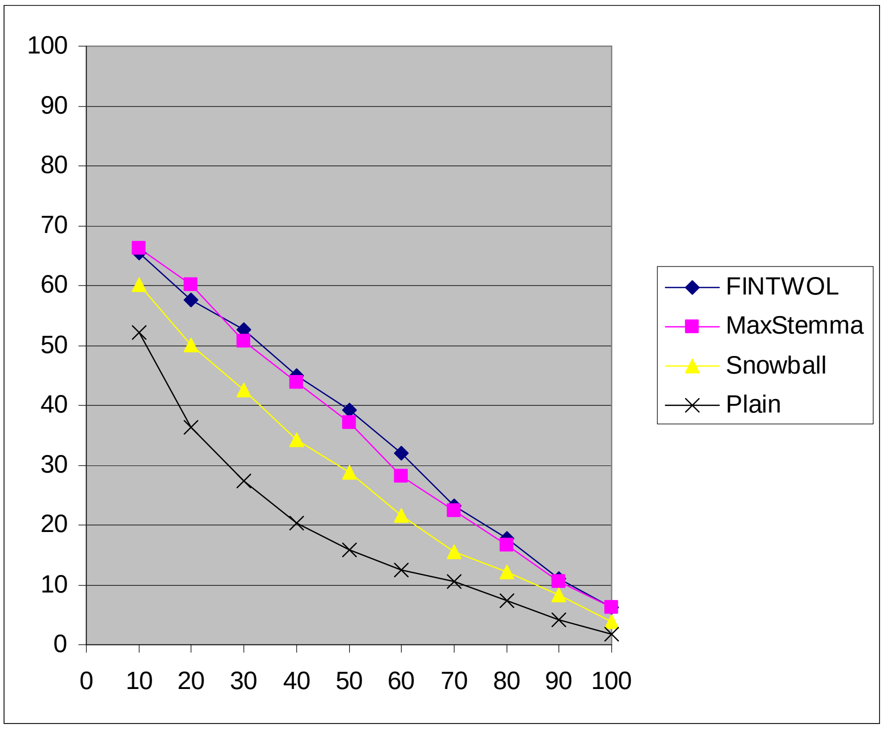


Figure 3. P-R curves on 10 recall levels on the normal relevance level in Environment One.

On the normal relevance level MaxStemma starts better than FINTWOL, but after the recall level 30 FINTWOL is a few per cent better down to recall level 70, where the curves start to merge again. Snowball performs clearly below FINTWOL and MaxStemma, but its performance is again quite consistent and smooth. Plain Words get performances that are ca. 45 – 79 % of FINTWOL's until recall level 70, where the relative difference starts to decrease. On average Plain words achieve 54 % of FINTWOL's performance level.

Results of FINTWOL and MaxStemma query-by-query on stringent and normal relevance levels are shown in Figures 4 and 5.

Figure 4. Results of FINTWOL and MaxStemma query-by-query on the stringent relevance level in Environment One.

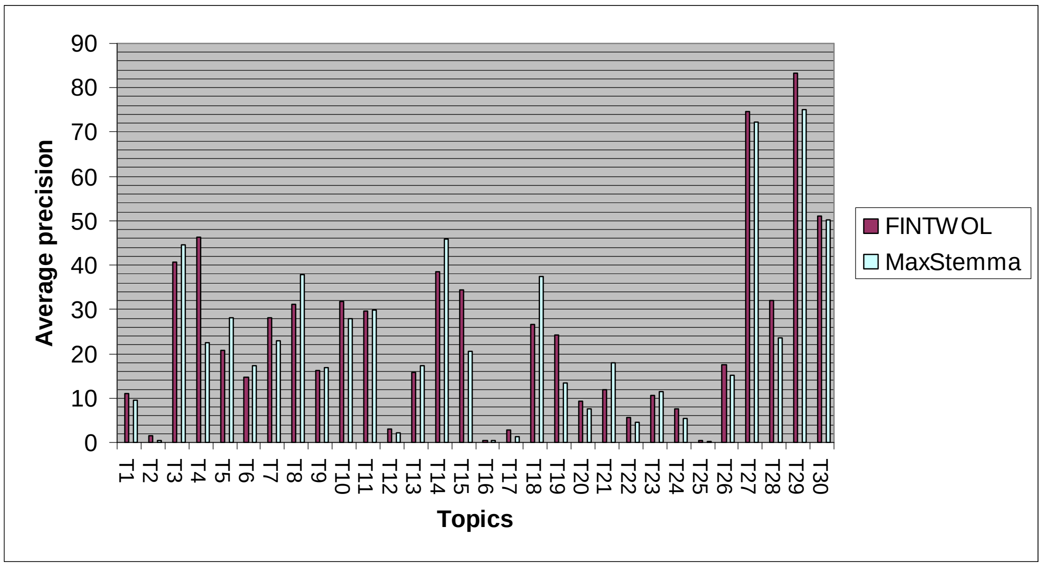
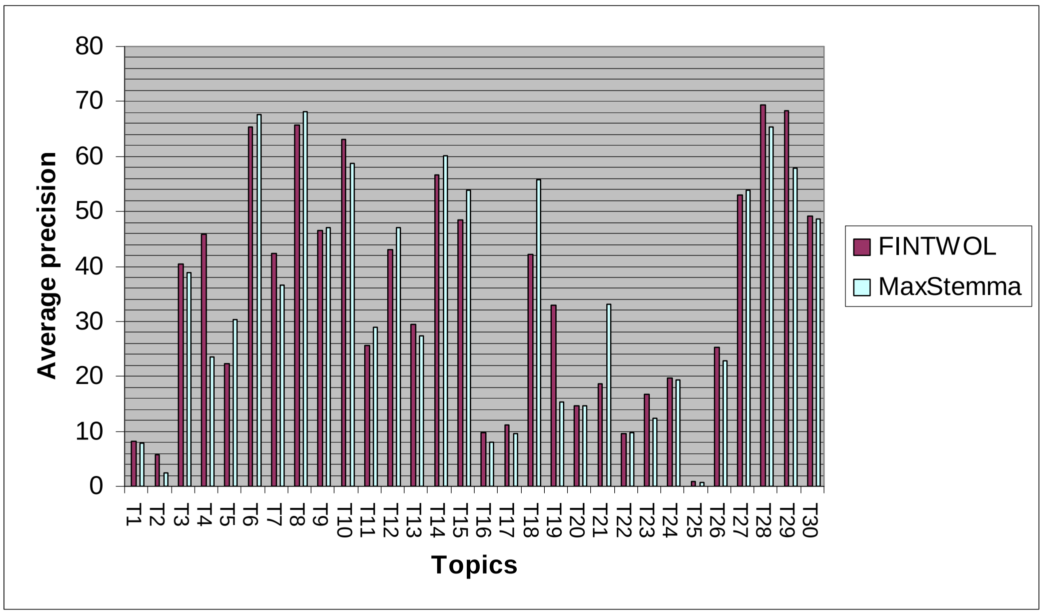


Figure 5. Results of FINTWOL and MaxStemma query-by-query on the normal relevance level in Environment One.



The results of the query-by-query analysis are mostly on a par between FINTWOL and MaxStemma. The largest differences on stringent relevance level in favor of FINTWOL are on topics T4, T7, T15, T19, T28, and T29. On topics T5, T8, T14, T18, and T21 MaxStemma outperforms FINTWOL clearly.

On normal relevance level FINTWOL outperforms MaxStemma clearly on topics T2, T4, T7, T19, and T29. MaxStemma gains clearly better results on topics T5, T15, T18, and T21.

Kunttu (2003, pp. 67 – 70) has explained part of the differences of results between FINTWOL and Finstems in Environment Two as products of different handling of compound sign ("-") in different types of indexes (morphologically analyzed, compounds as whole - LEMNS1 and LEMNS2 - and inflected index - INFL1 and INFL2). This seems to be the most obvious reason for some of the query-by-query results of Environment One also. At least this affects the results of query T2, T28, and T29. Otherwise no obvious reason for differences was found in Environment One, when results per query were compared.

Test results in Environment Two

Part of the results of Environment Two are shown in this section. Our main interest is in showing results that are either different from those in Environment One or that were not tested at all in Environment One. Main results of Environment Two are shown in Table VII.

Table VII. Average precision results on the stringent, normal and liberal relevance levels in Environment Two; absolute differences between methods in percentage units.

Stringent	**LEMS1**	**LEMS2**	**LEMNS1**	**LEMNS2**	**INFL1**	**INFL2**
Avg. precision over recall levels (%)	22,8	23,4 (0,6)	21,6 (-1,2)	22,0 (-0,8)	20,6 (-2,2)	21,1 (-1,7)

Normal	**LEMS1**	**LEMS2**	**LEMNS1**	**LEMNS2**	**INFL1**	**INFL2**
Avg. precision over recall levels (%)	33,0	33,7 (0,7)	30,3 (-2,7)	30,6 (-2,4)	29,7 (-3,3)	29,6 (-3,4)

Liberal	**LEMS1**	**LEMS2**	**LEMNS1**	**LEMNS2**	**INFL1**	**INFL2**
Avg. precision over recall levels (%)	34,3	35,1 (0,8)	31,3 (-3,0)	31,9 (-2,4)	32,1 (-2,2)	32,1 (-2,2)

In Environment Two, FINTWOL with indexes where compounds were split (LEMS1, LEMS2, cf. Table V for explanation on different method and index types) were the two best environments. The best results were reached with derivational queries (LEMS2) on all relevance levels, the difference to basic queries (LEMS1) varying between 0.6 − 0.8 % units. Differences of LEMS1 and LEMS2 with INFL1 and INFL2 varied between 1.7 − 4.1 % units in favor of LEMS1 and LEMS2.

When FINTWOL with compounds-as-whole (LEMNS1 and LEMNS2) was compared with Finstems on the liberal relevance level, the difference between the programs was maximally 0.8 % units in favor of Finstems. On the normal relevance level, again, FINTWOL performed slightly better, and the difference between the programs was maximally 1.0 % units in favor of

FINTWOL. On stringent relevance level, FINTWOL performed slightly better, too, and the difference between the programs was maximally 1.4 % units in favor of FINTWOL.

Some differences were found, when derivational queries were compared with normal queries in Environment Two. With FINTWOL, derivational queries performed slightly better than basic queries, and the difference varied between 0.3 − 0.8 % units in favor of derivational queries. With Finstems the maximal difference between INFL1 and INFL2 was 0.5 % units in favor of derivational queries, but mostly there was no difference at all.

In a more detailed analysis of topics by their content, four groups of topics were established: topical queries, geographically oriented queries, person oriented queries and organizational queries. These queries were analyzed only on normal relevance level. Best results were gained with person oriented queries, where the best average precision was in LEMNS1, 45.2 %.

Statistical testing of the differences

The statistical testing of the differences between methods was done using the Friedman test (original Friedman test, cf. Siegel and Castellan, 1988, modifications used in here in Conover, 1980). All used methods were evaluated against each other with different relevance levels. Significant differences at levels 0.001 and 0.01 in Environment One are shown in Table VIII.

Table VIII. Differences between methods significant at levels 0.001 and 0.01 using the Friedman test. Differences at 0.001 level shown **in bold**. Testing Environment One.

Comparison	**Significant difference**
FINTWOL > MaxStemma	None on any relevance level
FINTWOL > Snowball	Stringent**, Normal**
FINTWOL > Plain Words	**Stringent, Normal**
MaxStemma > Snowball	**Normal**
MaxStemma > Plain Words	**Stringent, Normal**
Snowball > Plain Words	Stringent, **Normal**

In Table IX statistically significant differences between methods in Environment Two are shown.

Table IX. Statistically significant differences of methods in Environment Two. Almost significant (p ≤ 0,05), significant (p ≤ 0,01), very significant (p ≤ 0,001)

	Liberal relevance	**Normal relevance**	**Stringent relevance**
LEMS1 > LEMNS1	very significant	---	---
LEMS1 > INFL1	---	---	almost significant
LEMS2 > LEMNS1	very significant	almost significant	---
LEMS2 > LEMNS2	significant	---	---
LEMS2 > INFL1	significant	significant	significant
LEMS2 > INFL2	almost significant	---	---
LEMNS1 > LEMNS2	almost significant	---	---
LEMNS1 > INFL2	significant	---	---
LEMNS2 > INFL1	---	significant	---

In the more detailed analysis of topics, only geographically oriented queries and organizational queries had statistically significant differences. Differences are summed in table X.

Table X. Statistically significant differences between topically analyzed queries in Environment Two. Normal relevance level.

Normal relevance	**Geographically oriented queries**	**Organizational queries**
LEMS1 > LEMNS1	almost significant	---
LEMS1 > LEMNS2	---	significant
LEMS1 > INFL1	almost significant	---
LEMS1 > INFL2	---	significant
LEMS2 > LEMNS1	very significant	---
LEMS2 > LEMNS2	---	almost significant
LEMS2 > INFL1	very significant	---
LEMS2 > INFL2	almost significant	almost significant
LEMNS2 > INFL1	---	almost significant
INFL1 > INFL2	---	almost significant

Impact of the approach on query times

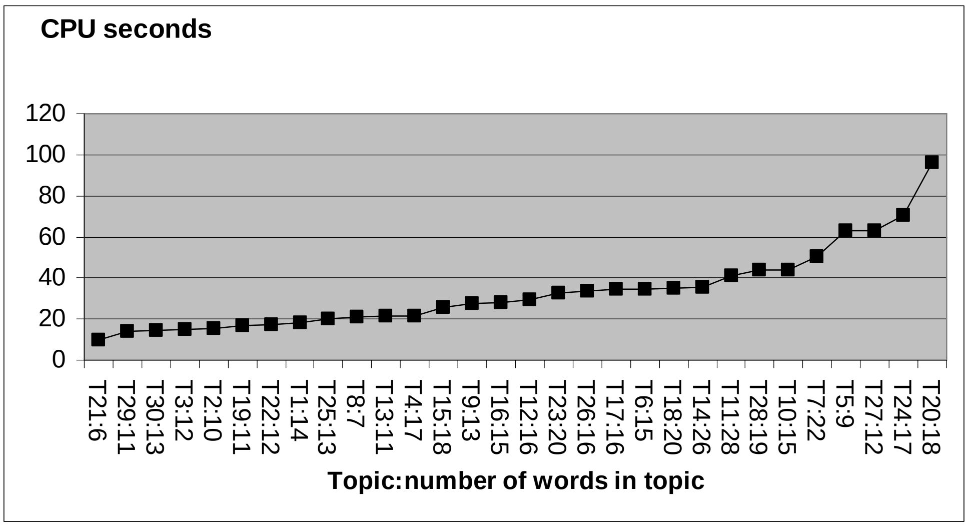
In *The query process* section we mentioned that when the stem generation procedure is used with INQUERY, search key truncation has to be simulated, because it is not supported in INQUERY. Grepping of the index entries with inflectional stems is used for this purpose.

Because all best match engines do not support search key truncation, and because the approach of matching the inflectional stems to the index in order to simulate truncation is potentially slow to compute, we wanted to analyze the run times of the simulated truncated queries. This does not provide direct efficiency data for a situation where truncation is actually implemented within a best match engine - the present simulation is certainly less efficient, slower. However, if the run times are tolerable even under the present test conditions, then they would be better, and thus acceptable, when properly implemented and integrated in the retrieval engine. After all, the query complexity (its number of keys) is a determining factor of processing times and (simulated) truncation tends to produce excessive number of keys.

When using the proposed method, the queries tend to become long, because all matching strings in the index are taken into the query. Final queries in Environment One have 934 − 24 443 query words. This is a potential bottleneck and problem of the process. The mean running times of queries of 6 − 28 source words were 10 − 96 CPU seconds of system time on a Sun Sparc Station (two 1,015 GHz processors, 4 GB memory) under a timesharing load of a few concurrent users. The effective user response time varied from 20.6 seconds to 4.15 minutes. Most of the time in the process is spent on grepping of the index. Unnecessary time in the process was also spent on screen output; without it the response times would be faster.

Results of the CPU time tests are shown in Figure 6 (mean CPU seconds of three consecutive runs, **system time + user time** of Unix's *time* function added together).

Figure 6. Mean CPU times of queries in seconds with MaxStemma and index grepping, when system and user time are added together.



The proposed method is thus not very fast, but acceptable for a short query, remembering that the test topics of the TUTK collection are quite long and result in long queries in an automated query generation process. Short queries of a few words, such as queries in the web (e.g., Jansen and others, 2000), would not cause considerable slowness in the process. When ad hoc short versions of TUTK topics with 3 – 5 words were tried out, run times decreased strongly.

Discussion

Three main research problems were formulated for this study. Firstly, we were interested to see how lemmatization and inflectional stem generation compare in a best-match environment. The second question was whether a stemmer is a realistic alternative for the handling of Finnish morphology for IR. Thirdly, the feasibility of simulation of truncation in a best-match system was raised. As sub-problems compound splitting, derivational queries, and different types of topics were studied.

For the first question it was shown, that lemmatization performed maximally 1.5 percentage units better than stem generation on average, when FINTWOL and MaxStemma were compared in Environment One. In Environment Two lemmatization with compounds-as-whole performed maximally 1.4 percentage units better than stem generation on average. Differences between lemmatization and stem generation were greatest, when lemmatization was accompanied with the use of split compound index and derivational queries in Environment Two. Thus it can be concluded that inflectional stem generation is a feasible method for covering the morphological variation of search keys in a best-match system.

Sparck Jones (1974) introduced measures for practical comparison of the importance of statistical differences between methods. If differences between two methods are statistically significant, their practical differences can be evaluated as a rule of thumb followingly:

- If the difference between methods is less than 5 % units, the practical difference is not noticeable.
- If the difference between methods is 5 – 10 % units, the practical difference is noticeable.
- If the difference between methods is >10 % units, the practical difference is material.

Differences between FINTWOL and MaxStemma were not statistically significant and their practical differences are not noticeable on any relevance level. The differences between average precisions on stringent level were not noticeable between FINTWOL and Snowball and MaxStemma and Snowball. On normal relevance level differences between FINTWOL and Snowball and MaxStemma and Snowball were noticeable (7.3 % units and 6.5 % units, respectively).

Differences between FINTWOL and plain words and MaxStemma and plain words were material on stringent relevance level. On the same relevance level the difference between the stemmer and plain words was noticeable.

On normal relevance level, the differences between FINTWOL and plain words and MaxStemma and plain words were material. The difference between Snowball and plain words on this relevance level was noticeable.

In Environment Two differences between FINTWOL and Finstems were not noticeable on any relevance level. In a more detailed analysis of topics by their content, only geographically oriented queries had noticeable differences between LEMS2 and INFL1 and LEMS2 and INFL2.

Is a stemmer, then, a realistic alternative to handling Finnish morphology for IR? Mayfield and McNamee (2003) report their experiments on CLEF 2002 collection with eight different languages. The languages include also Finnish and their tested methods include the Snowball stemmer, plain words and different types of n-grams. They get their best results for Finnish with 4-grams. Snowball works worse than 4-grams, but it is the second best method for Finnish and the stemmer gets over 10 % units better average results than in our study. However, the findings are not comparable across test collections.

Tomlinson (2003) shows results for nine languages at CLEF 2003. His basic comparison is between lexical and algorithmic stemming. The retrieval system used is based on the vector space model (Tomlinson, 2002, 2003). For Finnish the best results were achieved with the lexical stemmer using full lexicons, which are not described in more detail. Algorithmic stemming (Snowball) performs about 5 − 15 % units worse than lexical stemming, but clearly better than no stemming at all (difference varying from ca. 3 − 14 % units between Snowball and no stemming).

Airio and others (2003) studied the use of Snowball stemmer as an indexing method in their

CLEF experiment. Their conclusion was that indexing with Snowball was not a good method for CLIR purposes. The performance of the stemmed indexes of Finnish material was about 15 % units lower than results with morphological lemmatization.

Earlier stemming has not been really used for Finnish IR. Considering Snowball's origin and stemming style (suffix stripping, which is not very suitable for a heavily inflected language, cf. Frakes, 1992, Pirkola, 2001) its performance is better than expected although clearly below the performance of lemmatization and stem generation. It is possible, that a well-done linguistically motivated stemmer could be enough for handling the morphology of Finnish for IR purposes. For example in Kraaij and Pohlman (1996) and Braschler and Ripplinger (2003) several different types of stemmers are introduced for Dutch and German and some of them are fairly sophisticated, while their analysis is based on large dictionaries and is also linguistically motivated. Such a program is also Krovetz' stemmer (2000). It is thus reasonable to assume that a more sophisticated stemmer for Finnish would also perform better. However, the demarcation line between a stemmer and a lemmatizer would be hard to draw if the stemmer is using full dictionaries, as mentioned in the *Methods* section.

A baseline of plain words was also used in this study. In Mayfield and McNamee's (2003) study plain words get performance levels that are over 20 % on average, which is about 63 % of the best average result with n-grams. Tomlinson (2003) achieves as high average precision as 30.1 % with plain words, which is about 54 % of the best average results. Our best average precision for plain words was 18.9 % on normal relevance level. In a morphologically complex language such as Finnish even these levels sound somehow surprising. One may assume, that the explanation of this phenomenon is a known linguistic property of texts. Although Finnish nouns have theoretically some 2200 different forms, only a fraction of them occur in running text (Karlsson, 1986, 1983). Most of the existing forms are the same case forms, usually the so called grammatical function cases. Different case forms occur also due to the semantics of the word: e.g. words denoting to places occur mostly in nominative and locative cases etc. Usually the base word form and some case forms of it occur in the same texts. Thus even a plain unprocessed topic word given as a query word gives reasonable query results. Also the conjunctional effect of a set of query words may be an explaining factor; the sum effect of the different query words tends to filter out irrelevant hits and boost performance somewhat.

We can also compare these methods on a more general level. Three kinds of benefits are usually associated with different types of morphological processing of words in IR (Harman, 1991). Shortly put they are as follows:

- ease of use for the user (the morphology of query words is taken care of by the retrieval system),
- storage savings (smaller indexes when lemmatization or stemming is used), and
- improved retrieval performance.

Table XI compares different methods of the present study along these dimensions. It depends pretty much on one's preferences, which factor should be given most salience. If one graded these factors so that improved retrieval would be the most important (3 points), ease of use second (2) and storage savings third (1), we would get the first score in the SUM column. Another score (in parentheses) is based on grading ease of use (3), improved retrieval (2) and storage savings (1).

Table XI. Benefits of different methods compared

Method	Ease of use	storage savings	improved retrieval performance	**Sum**
lemmatization	Yes	Yes	Yes	6 (6)
stem generation	Yes	No	Yes	5 (5)
stemming	Yes	Yes	No?	3 (4)
plain words	Yes	No	No	2 (3)

On a whole, lemmatization gets the best score. Factors diminishing the usability of lemmatization can be also listed, and they are:

- need of a large lexicon, that needs updating,
- unknown words that cannot be processed while they are not in the dictionary of the lemmatizer (e.g. different types of proper names), and
- a longer implementation time, if the language does not already have a lemmatizer (compared with implementation of a stem generation and stemming program).

Lemmatizing performs somewhat better than stem generation, and it uses smaller indexes, runs faster and does not use system resources as much as the stem process with the rather resource consuming simulation of keyword truncation. Still stem generation has its benefits:

the implementation of a stem generation program with no large lexicons is quite easy and fast. Making of the indexes is also straightforward: no special index entry runs are needed, as texts can be indexed as running words.

Thirdly, the simulation of truncation seems to be feasible in a best-match system. Although the topics of TUTK were rather long and resulted in very long queries, runtimes were still manageable, at least for a laboratory system. The gain achieved by longer response times is however 9.2 % units in average precision – the average difference between the average precision readings of the proposed method and using just plain words. Moreover, the performance is virtually as good as may be achieved by lemmatization. In a realistic query system, real user queries would not be this long and should not cause serious trouble. However, there would be many queries at the same time and this may be resource consuming.

Conclusion

We have tested three different methods for handling the morphological variation of Finnish query words in a probabilistic IR environment. Also, queries with totally unprocessed query words were tried out.

The main results of this study were the following:

- Differences between stem generation and lemmatization were small and not statistically significant in Environment One; their practical differences were not noticeable on any relevance level. Only in individual queries there were greater differences.
- When lemmatization was enhanced with index using split compounds, best overall results were reached in Environment Two.
- The stemmer for Finnish performed reasonably well, although the stemmer was not

optimal. Still the difference between the stemmer and lemmatization or stem generation was mostly statistically and practically significant.

- Plain unprocessed query words delivered a performance in the range of 48 to 58 % of the lemmatizer's performance which is more than one might expect due to the highly inflectional language (Finnish) used here; but still their use is not realistic for Finnish.

[1] FINTWOL is an implementation of the two-level model for Finnish by Lingsoft (http://www.lingsoft.fi), its original contribution is (Koskenniemi, 1983). Finstems derives from Koskenniemi (1985) and its present implementation is by Lingsoft.

[2]Snowball (Porter, 2001) is a language for defining stemmers. A stemmer for Finnish has been produced according to its ideas and based on linguistic description of Finnish. Algorithm of the stemmer is described on the web page *The Finnish stemming algorithm* (cf. references).

[3] But as Porter (2001) mentions, stemming approaches are seldom purely algorithmic or dictionary-based. Usually even simple stemmers include exception lists “that are effectively mini-dictionaries”.

Acknowledgements

Thanks to Ms. Eija Airio from the Dept. of Information Studies at the University of Tampere for implementing the Unix scripts which automatize the query process in Environment One. Ph.D. Heikki Hyyrö from the Dept. of Computer Science at the University of Tampere and Mr. Heikki Keskustalo from the Dept. of Information Studies at the University of Tampere gave good hints on using the *time* command of Unix. We are also grateful to the FIRE research group for helpful comments.

The InQuery search engine was provided by the Center for Intelligent Information Retrieval

at the University of Massachusetts, Amherst.

FINTWOL (morphological description of Finnish). Copyright © Kimmo Koskenniemi and Lingsoft plc. 1983 – 1993.

Finstems Copyright © Kimmo Koskenniemi and Lingsoft plc. 1983 – 1993.

References

Abu-Salem, H., Al-Omari, M. and Evens, M. (1999), “Stemming Methodologies Over Individual Query Words for an Arabic Information Retrieval System”, *Journal of the American Society for Information Science*, Vol. 50 No. 6, pp. 524 − 529.

Airio, E., Keskustalo, H., Hedlund T. and Pirkola, A. (2003), “Multilingual Experiments of UTA at CLEF 2003. The impact of different merging strategies and word normalizing tools”, in Peters, C. and Borri, F. (eds.) *Results of the CLEF 2003 Evaluation Campaign*, Cross-Language Evaluation Forum, Italy, pp. 13 − 18.

Alemayehu, N. and Willet, P. (2003), “The effectiveness of stemming for information retrieval in Amharic”, *Program*, Vol. 37 No. 4, pp. 254 − 259.

Alkula, R. (2000), *Merkkijonoista suomen kielen sanoiksi.* Acta Universitatis Tamperensis 763, available at http://acta.uta.fi/pdf/951-44-4886-3.pdf (accessed March 31st 2004).

Alkula, R. (2001), “From Plain Character Strings to Meaningful Words: Producing Better Full Text Databases for Inflectional and Compounding Languages with Morphological Analysis Software”, *Information Retrieval*, Vol. 4, pp. 195 − 208.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999), *Modern information retrieval*, ACM Press, New York.

Blåberg, O. 1994. *The Ment Model – Complex States in Finite State Morphology*, Reports from Uppsala University, Dept. of Linguistics. RUUL 27.

Braschler, M. and Ripplinger, B. (2003), “Stemming and Decompounding for German Text Retrieval”, in *Advances in Information Retrieval*. 25th European Conference on IR Research, ECIR 2003, Pisa, Italy, pp. 177 − 192.

Broglio, J., Callan, J., Croft, B. and Nachbar, D. (1995), “Document Retrieval and Routing Using the INQUERY System”, in *Proceedings of the Third Text Retrieval Conference (TREC-3*), Gaithesburg, MD: National Institute of Standards and Technology, special publication 500-225, pp. 29 − 38.

Callan, J., Croft, B. and Harding, S. (1992), "The INQUERY Retrieval System", in *Proceedings of the Third International Conference on Databases and Expert Systems Applications*, Berlin: Springer Verlag, pp. 78 − 84.

Conover, W.J. (1980), *Practical Nonparametric Statistics,* 2nd edition, John Wiley and Sons, New York.

Frakes, W. (1992), "Stemming Algorithms", in Frakes, W. and Baeza-Yates, R. (eds.), *Information Retrieval. Data Structures and Algorithms,* Prentice Hall, pp. 131 − 160.

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

Harman, D. (1991), "How effective is Suffixing?", *Journal of the American Society for Information Science,* Vol. 42 No. 1, pp. 7 − 15.

Hull, D. (1996), "Stemming Algorithms: a Case Study for Detailed Evaluation", *Journal of the American Society for Information Science,* Vol. 47 No. 1, pp. 70 − 84.

Jacquemin, C. and Tzoukerman, E. (1999), "NLP for term variant extraction: synergy between morphology, lexicon, and syntax", in Strzalkowski, T. (ed.), *Natural Language Information Retrieval,* Kluwer Academic Publishers, Dordrecht, pp. 25 – 74.

Jansen, B., Spink, A. and Sarasevic, T. (2000), "Real life, real users, and real needs: a study and analysis of user queries on the web", *Information Processing and Management,* Vol. 36, pp. 207 – 227.

Jäppinen, H. and Ylilammi, M. (1986), "Associative Model of Morphological Analysis: an Empirical Inquiry", *Computational Linguistics,* Vol. 12 No. 4, pp. 257 – 272.

Karlsson, F. (1983), *Suomen kielen äänne- ja muotorakenne*. Wsoy, Helsinki.

Karlsson, F. (1986), "Frequency Considerations in Morphology", *Zeitsschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung,* Vol. 39 No. 1, pp. 19 – 28.

Kekäläinen, J. (1999), *The Effects of Query Complexity, Expansion and Structure on Retrieval Performance in Probabilistic Retrieval*. Acta Universitatis Tamperensis 678.

Kettunen, K. (1991a), "Doing the Stem Generation with Stemma", in J. Niemi (ed.), *Papers from the Eighteenth Finnish Conference of Linguistics*. Kielitieteellisiä tutkimuksia, Joensuun yliopisto, N:o 24, 1991, pp. 80 – 97

Kettunen, K. (1991b), "Stemma, a Robust Noun Stem Generator for Finnish", *Humanistiske Data* No. 1, pp. 26 – 31.

Koskenniemi, K. (1985), "FINSTEMS: a Module for Information Retrieval", in Karlsson, F.

(ed.), *Computational morphosyntax*. Report on research 1981 - 84. Publications of the Department of General linguistics, University of Helsinki. No. 13., pp. 81 – 92.

Koskenniemi, K. (1983), *Two-Level Morphology: a General Computational Model for Word-form Recognition and Production*. Publications of the Department of General linguistics, University of Helsinki. No. 11.

Kraaij, W. and Pohlmann, R. (1996), "Viewing stemming as recall enhancement", in Frei, H.-P., Harman, D., Schäuble, P. and Wilkinson, R. (eds.) *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (SIGIR '96), ACM Press, pp. 40 − 48.

Krovetz, R. (2000), "Viewing morphology as an inference process", *Artificial intelligence*, Vol. 118, pp. 277 − 294.

Kunttu, T. (2003), *Perus- ja taivutusmuotohakemiston tuloksellisuus todennäköisyyksiin perustuvassa tiedonhakujärjestelmässä*. Informaatiotutkimuksen pro gradu -tutkielma. Informaatiotutkimuksen laitos, Tampereen yliopisto. (M.Sc. Thesis, University of Tampere, Dept. of Information Studies).

Lovins, J. (1968), "Development of a stemming algorithm", *Mechanical Translation and Computational Linguistics*, Vol. 11, pp. 22 − 31.

Matthews, P. H. (1991), *Morphology*, Second edition, Cambridge University Press.

Mayfield, J. and McNamee, P. (2003), "Single N-gram Stemming", in *Proceedings of Sigir2003*, The Twenty-Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 415 − 416.

Pirkola, A. (2001), "Morphological Typology of Languages for IR", *Journal of Documentation*, Vol. 57 No. 3, pp. 330 − 348.

Popovič, M. and Willet, P. (1992), "The effectiveness of Stemming for Natural-Language Access to Slovene Textual Data", *Journal of the American Society for Information Science*, Vol. 43 No. 5, pp. 384 − 390.

Porter, M. (1980), "An Algorithm for Suffix Stripping", *Program*, Vol. 14, pp. 130 − 137.

Porter, M. (2001), "Snowball: A language for stemming algorithms", available at http://snowball.tartarus.org/texts/introduction.html (accessed Nov. 28, 2003).

Schinke, R., Greengrass, M., Robertson, A. and Willet, P. (1996), "A Stemming algorithm for Latin text databases", *Journal of Documentation*, Vol. 52 No. 2, pp. 172 − 187.

Sever, H. and Bitirim, Y. (2003), "FindStem: Analysis and Evaluation of a Turkish Stemming Algorithm", in Nascimento, M., Moura, E. and Oliveira, A. (eds.), *String Processing and*

Information Retrieval, 10th International Symposium, SPIRE 2003, pp. 238 − 251.

Siegel, S. and Castellan, J. Jr. (1988), *Nonparametric statistics for the behavioural sciences*, McGraw-Hill, New York.

Silva, G. and Oliveira, C. (2003), “The Implementation and Evaluation of a Lexicon-Based Stemmer”, in Nascimento, M. Moura, E. and Oliveira, A. (eds.), *String Processing and Information Retrieval*, 10th International Symposium, SPIRE 2003, pp. 266 − 276.

Sormunen, E. (2000), *A Method for Measuring Wide Range Performance of Boolean Queries in Full-Text Databases.* Acta Universitatis Tamperensis 748.

Sparck Jones, K. (1974), “Automatic indexing”, *Journal of Documentation*, Volume 30 No. 4, pp. 393 − 432.

The Finnish stemming algorithm, available at http://snowball.tartarus.org/finnish/stemmer.html (accessed Nov. 28., 2003).

Tomlinson, S. (2002), “Experiments in 8 European Languages with Hummingbird SearchServer™ at CLEF 2002”, available at http://clef.iei.pi.cnr.it:2002/workshop2002/WN/26.pdf (accessed April 28, 2004).

Tomlinson, S. (2003), “Lexical and Algorithmic Stemming Compared for 9 European Languages with Hummingbird SearchServer™ at CLEF 2003”, available at http://clef.iei.pi.cnr.it/2003/WN_web/19.pdf (accessed April 28, 2004).