Effects of Stemming on Turkish Text Retrieval

by

Ayşın Solak
and
Fazlı Can

BU-CEIS-94-20

DEPARTMENT
of
COMPUTER ENGINEERING
and
INFORMATION SCIENCES

BILKENT UNIVERSITY

Effects of Stemming on Turkish Text Retrieval

Ayşın Solak and Fazlı Can

Bilkent University, Bilkent, 06533 Ankara, Turkey

Abstract

A stemming algorithm developed for the Turkish language is presented. The retrieval performance of the algorithm is then measured in terms of its effect on retrieval performance of a best-match text retrieval system. The results show that the stemming algorithm reduces the index dictionary size significantly and also improves the retrieval effectiveness.

1 Introduction

The widespread availability of text in machine-readable form has led to increasing use of *text retrieval* systems that are designed for the storage, retrieval and processing of textual databases. Such systems allow a user to specify a query by a set of natural language words; the database is scanned for those documents that match the query and these are presented to the user. Many of such systems have been based on the use of boolean searching, where the user must specify AND, OR, or NOT logical operators between the words of the query; however, the new trend is the use of *nearest neighbour*, or *best-match* retrieval systems, where the user specifies just a natural language query without any logical operators. In this case, the system ranks the documents in the order of decreasing similarity with the query [13]. Such systems make extensive use of *automatic indexing* techniques, rather than relying on manual indexing. An automatic indexing procedure processes an input text string, either a document or a query, in two stages [10]:

1. Each word in the input string is compared with a *stopword list*, a dictionary of high frequency words, such as AND, OF, THE, TO, etc. in English. Removal of such words provides an effective way of reducing the storage requirements of the document and query representatives. Words from the input string that are not on the stopword list are assumed to be descriptive of the document or query.

2. The remaining words are then passed on for processing by a *stemming algorithm*, a computational procedure that conflates morphological variants of a given root to a single stem by removing (or sometimes replacing) the inflectional and derivational suffixes. The resulting word stems are then used to characterize the document or query for indexing purposes.

Many such text retrieval systems have been implemented. Among them are MEDLINE, implemented at the National Library of Medicine; BRS, developed by Bibliographic Re-

trieval Services; ORBIT, developed by the System Development Corporation; and DIALOG, designed by Lockheed Information Systems [12]. However, the great majority of such works have been carried out with English language material, where stopword lists and stemming routines have been available for many years. In this study, we aimed to evaluate the use of stemming and best-match searching in Turkish text retrieval. Using a stemming algorithm that takes great account of the language's morphological structure, a database of Turkish news and a set of Turkish queries are indexed automatically. The performance of the stemming algorithm is then measured in terms of its effect on retrieval performance of the system.

The structure of the paper is as follows: Section 2 presents an overwiev of the concept of stemming in information retrieval. A brief information about the general properties of the Turkish language and the details of the stemming algorithm used in this study are given in Section 3. Section 4 covers the characteristics of the databases and queries used in this work, while Section 5 presents the method applied to generate the stopword lists. In Section 6, an overwiew of the best-match retrieval method is presented. Last two sections provide the retrieval effectiveness results and the conclusions obtained in our study, and give some suggestions about the future work.

2 Stemming

One technique for improving information retrieval performance is to provide morphological variants of query terms [4]. For example, if the user enters the term COMPILER as part of a query, it is likely that s/he will also be interested in such variants as COMPILE, COMPILED, COMPILATION, etc. In natural language processing, *conflation* is the process of matching morphological term variants. Conflation can either be manual, i.e., using some kind of regular expression, or automatic, via programs called *stemmers*. Stemming is also used in IR to reduce the size of index files.

There are several approaches to stemming [4]. Among those, the most popular one is called as *affix removal* stemming. Affix removal stemmers remove suffixes and/or prefixes from terms leaving a stem. A number of stemming algorithms have been described in the literature,[1] notable among them being Lovins [7], Dawson [3], Porter [11], and Paice [9]. However, all of these stemmers are specific to English text. Especially for agglutinative languages (in which words are formed by combining root words and morphemes, such as Japanese, Turkish, Finnish, Hungarian, etc.) there are limited number of known studies [5, 6]. Stemming in such languages are not so straightforward as in English because it is necessary to make significant phonological and morphological analyses.

There are several criteria for judging performance of a stemmer in IR: correctness, retrieval effectiveness, and compression performance. A stemming algorithm can be considered as incorrect if it causes overstemming (removal of too much of a term) or understemming (removal of too little of a term). Stemmers of IR are not usually judged on the basis of linguistic correctness [4]. The effect of a stemming algorithm on retrieval performance in IR test collections is usually measured in terms of recall and precision.[2]

[1] For other sources see Frakes [4].

[2] Recall is the proportion of relevant documents retrieved, while precision is the proportion of retrieved documents that are relevant.

Stemmers can also be rated on their compression performance.

In Frakes [4], various studies of stemming in English for improving retrieval effectiveness and their results can be found. Those studies prove that stemming affects retrieval performance positively. There is no evidence that stemming will degrade retrieval effectiveness. The results also show that there is a little difference between the retrieval effectiveness of different stemmers.

3 A Stemming Algorithm for Turkish

3.1 The Turkish Language

Turkish is an agglutinative language, which means that it expresses syntactic relations between words or concepts through discrete suffixes, each of which conveys a single idea. For an agglutinative language such as Turkish, the concept of word is much larger than the set of vocabulary items. Words can grow to be relatively long by addition of suffixes and sometimes contain an amount of semantic information equivalent to a complete sentence in another language. A popular example of complex Turkish word formation is

ÇEKOSLOVAKYALILAŞTIRAMADIKLARIMIZDANMIŞSINIZ

whose equivalent in English is "(it is speculated that) You had been one of those whom we could not convert to a Czechoslovakian." In this example, one word in Turkish corresponds to a full sentence in English. Each suffix has a certain function and modifies the semantic information in the stem preceding it.

Turkish word formation uses a number of phonetic harmony rules. Vowels and consonants change in certain ways when a suffix is appended to a stem, so that such harmony constraints are not violated.

Turkish roots can be classified into two main classes as *nominal* and *verbal* roots. The suffixes that can be received by either of these groups are different, i.e., a suffix which can be affixed to a nominal root can not be affixed to a verbal root with the same semantic function.

Turkish suffixes can be classified as *derivational* and *inflexional*. Derivational suffixes change the meaning and sometimes the class of the stems they are affixed to, while an inflected verb or noun remains as such after the affixation. Inflexional suffixes can be affixed to all of the roots in the class that they belong to. On the other hand, the number of roots that each derivational suffix can be affixed to changes.

The simplified models for nominal and verbal grammars can be given as follows:[3]

The nominal model:

nominal root + plural suffix + possessive suffix + case suffix + relative suffix

[3]Refer Solak [15] for detailed information on each of the suffixes in these models and the exceptional cases about them.

The verbal model:

verbal root + voice suffixes + negation suffix + compound verb suffix + main tense suffix + question suffix + second tense suffix + person suffix

3.2 Stemming Algorithm

The complexity of Turkish morphology, as summarized in the previous section, necessitates the use of a morphological analysis for the purpose of stemming. The stemming algorithm used in this study makes use of a morphological analyzer developed previously to be used for spelling checking of Turkish [15, 16, 17]. The documents and queries used in our experiments have been passed from the spelling checker and all the words in them are considered to be correctly spelled. As a preprocessing step, both the documents and queries are given as input to our stemmer and the stems of the words in both are written into separate files. After this, all the remaining operations are done on these stemmed database.

Morphological analysis of a Turkish word is handled in three steps:

1. Root determination,
2. Morphophonemic checks, and
3. Morphological parsing.

During these steps a dictionary of Turkish root words, and a set of rules for Turkish morphophonemics, and morphotactics are used concurrently.

3.2.1 Root Determination

Before parsing the morphological structures of a Turkish word, the root has to be determined. All parsers use a dictionary of correctly spelled words. A dictionary of all possible words is neither an efficient nor a practical approach. So, only root morphemes and some irregular stems are to be held in the dictionary. Our dictionary, of about 23,000 words, has been based on the Turkish Writing Guide (Türkçe Yazım Kılavuzu) [19, 20] as the source. Each entry of the dictionary contains a root word in Turkish and a series of flags showing certain properties of that word. We currently have reserved space for 64 different flags for a single word. If the bit corresponding to a certain flag is set for an entry then it means that the word which this entry belongs to has the property represented by that flag. Only 41 flags have been used yet, but later implementations may use the remaining ones.[4]

The root of a word is searched in the dictionary using a maximal match algorithm. In this algorithm, first the whole word is searched in the dictionary. If it is found then the word has no suffixes and therefore it does not need to be parsed. Otherwise, we remove

[4]The list of these flags together with some examples of root words for which those flags are applicable can be found in Solak [15].

a letter from the right and search the resulting substring. We continue this by removing letters from the right until we find a root. If no root can be found although the first letter of the word is reached, the word can not be stemmed and it is inserted into the stemmed database as is.

The maximum length substring of the word that is present in the dictionary is not always its root. If the word can not be parsed correctly using that root, a new root is searched in the dictionary, this time removing letters from the end of the previous root. If a new root can be found the same operations are repeated, otherwise the word is inserted into the stemmed database as is. For instance, the root of the word YAPILDIN (YOU WERE MADE) is first determined as the noun YAPI (STRUCTURE). However, the rest of the word does not form a valid sequence of suffixes for a nominal root. Instead of reporting YAPI as the root of the word, a new root is searched, and the verbal root YAP (MAKE, DO) is found. Since with this root the word can be parsed correctly, it is reported as the root of the word.

Root determination presents some difficulties when the root of the word is deformed. For the root words which have to be deformed during certain agglutinations (see [15, 17]), a flag indicating that property is set in the dictionary entry. The individual cases such as the dative and plural forms of personal pronouns are inserted into the dictionary and treated as exceptions. For the other root deformations, the root of the word is found by making some checks and some necessary changes.[5] For some roots both of the deformed and undeformed forms are valid. Such cases are handled again by the help of certain flags.

The exact root of a word can not be uniquely determined in some cases. A Turkish word may have more than one potential roots. For example, the root of the word KARIN might be KARIN (ABDOMEN), KARI (WOMAN, WIFE) or KAR (SNOW), giving the word the meanings ABDOMEN, YOUR WIFE or YOUR SNOW (or THE SNOW'S), respectively. Obviously, it is very difficult to determine which one is the exact root of the word without extensive semantic processing. For this reason, instead of reporting only one stem per a word, we prefered to find and add to the stemmed database all the possible roots of it. As a result, for some terms in the unstemmed database, there exists more than one terms (1.22 terms on the average) in the stemmed database.

3.2.2 Morphophonemic Checks

After a root of a word is found, the rest of the word is considered as the suffixes. Vowels and consonants within suffixes should obey certain rules during agglutination [15]. Therefore, the suffixes part of a word must be checked to see whether any of the morphophonemic rules are violated. The vowel harmony check may be done just after the root determination, but to perform the other morphophonemic checks, the suffixes must be determined. Because of this, these checks are done during morphological parsing, after each suffix is isolated. During the lexical analysis, if any of the allomorphs[6] of a suffix can be matched, it is sent to the parser without checking whether the correct form of it is used. These checks are done within the parser. Since the vowel harmony check is done beforehand, only the remaining morphophonemic checks must be done at that point. The

[5] In Solak [15, 17], some examples representing how the real value of a deformed root is determined can be found.

[6] An allomorph is any of the variant forms of a morpheme.

consonant harmony checks are among these checks [15].

If a word can not pass any of the morphophonemic checks, considering the possibility that the root may have been determined wrongly, a new root is searched in the dictionary, and the process is repeated.

3.2.3 Morphological Parsing

For the morphological parsing of Turkish words two separate sets of rules for the two main root classes have been prepared. When the root of a word is found, the class of the root determines which set of rules (i.e., verb parser or noun parser) is to be used for further parsing.

For the implementation of the lexical analyzers and parsers in which the rules are included, two standard UNIX utilities, *lex* and *yacc*, have been utilized respectively [8, 14]. Two sets of *lex* specifications, one per each root class, are prepared to generate the lexical analyzers which are to be called by the parsers each time a new token is needed. The specifications contain regular expressions that match suffix tokens. The lexical analyzer corresponding to the category of the current stem sends, as the next suffix token, the maximum length substring from the left of the remaining suffixes part that matches to any allomorph of a suffix in the permitted class.

The grammar rules for morphotactics of Turkish word structures have been described in two *yacc* specifications, one for nominal and one for verbal roots. The lexical analyzers produce the suffix token stream. *Yacc* generates the programs for the parsers. The two parsers are alternatively used. First parser to be used is determined according to the class of the root, but as the parsing continues it may be necessary to switch from one parser to another and continue there, or again pass back to the previous one, since the class of a stem can change when it receives certain suffixes. For example, while parsing continues in the noun parser, if the derivational suffix -L{A}Ş, which makes a verb from a noun (e.g., ŞAKALAŞMAK (to joke with another)), is matched, a jump to the verb parser must be done.

For the roots that can take all the suffixes belonging to both nominal or verbal classes, if parsing is unsuccessful in the first parser chosen, the other one must also be tried. For example, the root of the word AÇLAR (hungry people) is AÇ. This root may either be used as a verb (open) or as a noun (hungry). Parsing is first attempted with the verb parser, but it results unsuccessfully. So we backtrack and use the other parser. With the noun parser the word can be parsed successfully.

In current implementation, the grammar for verb parser consists of 230 rules in which 80 terminals and 81 nonterminals are used, and the grammar for noun parser, consists of 263 rules in which 68 terminals and 94 nonterminals are used.

4 Characteristics of Databases and Queries

In this study, a news database consisting of 533 Turkish documents is used as the text corpus. It is obtained from a Turkish news agency and contains news on different subjects

such as politics, sports, economics, etc. The queries are Turkish natural language queries prepared by graduate students.

4.1 Documents

Originally the news database included 110,605 words. After a preprocessing, all unnecessary symbols, numbers, etc. are removed from the database and only words which have a potential to become an index term are left. As an example, consider the following short document from the original database:

> İşçi emeklileriyle dul ve yetimlerinin maaş farkları bugün ödeniyor. Sosyal yardım zammı ile katsayının 1 temmuz 1992 tarihinde arttırmasından kaynaklanan 45 günlük maaş farkları ilgili banka şubelerince ödeniyor. 1 temmuz - 15 ağustos tarihlerini kapsayan maaş farkları toplamı 680 milyar lirayı buluyor. SSK'dan 1 milyon 790 bin emekli, dul ve yetim maaş alıyor.

After the preprocessing step, the above document looks like as follows:

> İşçi emeklileriyle dul ve yetimlerinin maaş farkları bugün ödeniyor Sosyal yardım zammı ile katsayının temmuz tarihinde arttırmasından kaynaklanan günlük maaş farkları ilgili banka şubelerince ödeniyor temmuz ağustos tarihlerini kapsayan maaş farkları toplamı milyar lirayı buluyor ssk'dan milyon bin emekli dul ve yetim maaş alıyor

The resulting database is used in the subsequent experiments and it is referred to as the *unstemmed database* in this report. It contains 90,912 words.

Applying the stemming routine on the unstemmed database, the stems of all the words in this database are found, and the *stemmed database* is created. The stemmed database contains 111,062 words. It contains more words than the unstemmed database, because, as explained in Section 3.2.1, for some words in the unstemmed database, the stemmer finds more than one root. Our example document looks like as follows in the stemmed database:

> İş emekli dul ve yetim maaş fark bugün öde Sosyal ya yar yardım zam il ile katsayı temmuz tarih tarihi art kaynaklan gün maaş fark ilgi ilgili bank banka şube öde temmuz ağustos tarih kapsa maaş fark topla toplam milyar lira bul bula ssk milyon bin emek emekli dul ve yetim maaş al

Using the stemmed and unstemmed databases two stopword lists are generated (see Section 5). The stopword list of the stemmed database contains 293 terms, where the other one contains 3128 terms. For both databases, the remaining unique terms are considered as the termlist to be indexed, i.e., the index dictionaries. The termlist for the stemmed database contains 6673 terms, while the other one contains 19086 terms. Hence, stemming decreases the index dictionary size to one third.

Both termlists are stored alphabetically and a term number is given to each term. Each document vector contains the <termno, weight> information for each unique term in that document. Table 1 lists the characteristics of both databases.

Database	No. of documents	No. of distinct terms	Avg. no. of distinct terms / doc	No. of terms in the stoplist
Stemmed	533	6673	78.13	293
Unstemmed	533	19086	86.85	3128

Table 1: Characteristics of the databases

Queries	No. of queries	No. of distinct terms	Avg. no. of distinct terms / query	Avg. no. of relevant doc.s / query	Total no. of distinct relevant doc.s
Stemmed	71	448	11.10	7.11	324
Unstemmed	71	521	11.23	7.11	324

Table 2: Characteristics of the queries

4.2 Queries

Originally 110 queries were written by seven graduate students. Then, the relevant documents for each query were determined. 39 queries with less than two relevant documents are eliminated, and remaining 71 queries are used in the experiments.

Queries are written in natural language form and they also contain a list of keywords to indicate the importance of certain terms in the query. They are written just in the same format as the original database. Applying the same preprocessing step, the queries are converted to a most suitable form, which is refered to as *unstemmed queries*. Afterwards, the *stemmed queries* are constructed just in the same way as the stemmed documents.

The query vectors are also prepared in the same way as document vectors. The only difference is that, if a query term is present in neither the termlist nor the stoplist, i.e., if it is not an index term, it is not represented in the query vector. The query characteristics are presented in Table 2.

5 Stopword List Generation

A stoplist is a list of words considered to have no indexing value, used to reduce the number of indexing terms. Each potential indexing term is checked against the stoplist and eliminated if found there [4]. The advantages of the process are not only that non-significant words are removed and will therefore not interfere during retrieval, but also that the size of the total document surrogate file can be reduced significantly.

In this study, two different stoplists are prepared, one to be used with the stemmed database, another one for the unstemmed database (see Section 4.1).

Term	English Meaning(s)	Frequency
VE	AND	2128
OL	BE	1917
BİR	ONE, A, AN	1784
BU	THIS	1347
DA	TOO, ALSO	1186
DE	TOO, ALSO, SAY	1072
ET	DO, MAKE, MEAT	1034
YAP	DO, MAKE	1030
İÇ	IN, INSIDE, DRINK	803
Mİ	Question suffix	729
İL	CITY	704
İÇİN	FOR	703
AL	TAKE, RED	671
İLE	WITH	663
YIL	YEAR	588
VER	GIVE	588
ARA	LOOK FOR, INTERVAL	520
BİL	KNOW	520
BUL	FIND	519

Table 3: Most frequently occuring terms in the stemmed database

5.1 Stemmed Stopword List

The stemmed database contains 6966 unique terms. Majority of these terms occur infrequently in the documents, where a small portion of them are very frequent. For example, while 2508 of these terms occur only once in the whole database, the term VE, meaning AND in English,occurs 2128 times. An inspection of the most frequently occuring terms, i.e., those that are primary candidates for inclusion in a stoplist, as seen in Table 3, confirms that they are mainly function words. Considering this property, the first stoplist is prepared by taking into account of the most frequently occuring terms.

It should be noted that there are some words in Turkish that have exactly the same written form but differ in their meaning. For example, as seen in Table 3, the term ET can be either the root form of the verbs TO DO or TO MAKE, or it can be used in themeaning MEAT. Hence, this term can be considered as a function word in the first usage, but not in the second. Since it is difficult to distinguish between the variants without extensive semantic processing, we prefered to include such words in the stoplist.

Another observation shows that some non-function words have a large frequency, e.g., the term İL in Table 3. This happens since sometimes the exact root of a word can not be determined uniquely, and there may be more than one term in the stemmed database corresponding to a single term of the unstemmed database (see Section 3.2.1). For example, the term İL seems to occur too much in the database, because for each occurence of the term İLE in the unstemmed database, both the terms İL and İLE are inserted into the stemmed database, since İLE can be used either in the meaning WITH

or it can be the dative form of the noun İL which means TO THE CITY. Such terms are also included in the stoplist.

The stopword list prepared according to the term frequencies contained only 200 terms and it was excluding some of the non-context bearing terms such as BUT, SO, THUS, ALTHOUGH in English. Although such termsdo not occur very frequently in the database, it is certain that theydo not have any indexing value. In order to determine such terms, a stopword list for English, given in [18], is examined and the corresponding Turkish terms are added to our stoplist. Consequently, the stopword list for the stemmed database includes 293terms. The terms in this stopword list cover nearly 40% of all word occurences.

5.2 Unstemmed Stopword List

In order to have equivalent conditions in both stemmed and unstemmed case, the stopword list of the unstemmed database is prepared by the help of the stemmed stoplist. A term in the unstemmed database is inserted into its stoplist only if all of its potential stems occur in the stemmed stoplist. For example, the term ALTINA is not included inthe unstemmed stoplist, because this term might have two meanings, TOTHE BOTTOM or TO GOLD, hence, two potential stems, ALT (BOTTOM) or ALTIN (GOLD). The first stem occurs in the stemmed stoplist, but the second one does not. Thus, the index for the stemmed database includes the term ALTIN, but not ALT. Therefore, the unstemmed database shouldalso include this term. On the other hand, the term ONDA, which can mean either AT TEN or AT HIM/HER/IT, is included in the unstemmed stoplist, because both of its potential stems, ON (TEN) and O (HE/SHE/IT), are members of the stemmed stoplist. Using this method, 3128 terms are inserted into the stopword list of unstemmed database, and they cover nearly 34% of all word occurences.

6 Best-Match Retrieval

In a *best-match retrieval* environment, the best-matching d_s documents out of m documents are retrieved [2]. There are several matching functions based on the term weighting used for document and query terms. For term weighting there are three components [13]:

1. **TFC** (Term Frequency Component)
2. **CFC** (Collection Frequency Component)
3. **NC** (Normalization Component)

The weights of the terms of a document and a query, denoted by w_{d_j} and w_{q_j}, $1 \leq j \leq n$, where n is the number of terms, are obtained by multiplying the respective weights of these three weighting components. After obtaining the term weights, the matching function for a query q and a document d is defined as the following vector product:

$$similarity(q,d) = \sum_{j=1}^{n} w_{q_j} * w_{d_j}$$

Abbreviation	TW1	TW2	TW3	TW4	TW5	TW6	TW7
Meaning	*txc.txx*	*tfc.nfx*	*tfc.tfx*	*tfc.bfx*	*nfc.nfx*	*nfc.tfx*	*nfc.bfx*

Table 4: Term weighting approaches

Consider d_{ij}. The weight of term t_j in d_i will be (TFC * CFC * NC). There are three possibilities for TFC: b, t and n, three possibilities for CFC: x, f and p, and two possibilities for NC: x and c. The meanings of those letters are as follows:

- **TFC**
  1. b is the binary weight where when a term is present the term frequency is ignored and TFC is taken as 1;
  2. t is raw term frequency tf, which means TFC is equal to the number of occurences of t_j in d_i;
  3. n is the augmented normalized term frequency, defined as
     $0.5 + \frac{0.5*tf}{maxtf}$
     where $maxtf$ is the maximum number of times any term appears in d_i.
- **CFC**
  1. x indicates no change, thus CFC is taken as 1;
  2. f is inverse document frequency, which is equal to
     $ln\frac{m}{t_{g_j}} + 1$
     where t_{g_j} indicates the number of documents containing t_j;
  3. p is the probabilistic inverse collection frequency factor, which is equal to
     $ln\frac{m-t_g+1}{t_g}$
- **NC**
  1. x means no normalization, thus NC is taken as 1;
  2. c is cosine normalization, which is equal to
     $\frac{1}{\sqrt{\sum w_i^2}}$
     The normalization of query terms is insignificant, since it does not change the relative ranking of documents.

The various combinations of the term weighting components yield different matching functions. For example, TW1, i.e., the combination *txc* (TFC = t, CFC = x, and NC = c) and *txx*, respectively, for documents and queries, yields the well known cosine function in IR literature [12, 18]. The combinations *tfc*, *nfc* for documents, and *nfx*, *tfx*, *bfx* for queries lead to better IR performance [13]. This gives us six different matching functions: *tfc.nfx*, *tfc.tfx*, *tfc.bfx*, *nfc.nfx*, *nfc.tfx* and *nfc.bfx*. Table 4 shows these seven combinations used in the experiments.

Effectiveness Measure		Q		Precision		Recall	
d_s		10	20	10	20	10	20
TW1	Unstemmed	10	6	0.258	0.170	0.501	0.628
	Stemmed	8	5	0.258	0.168	0.512	0.635
TW2	Unstemmed	6	4	0.280	0.182	0.552	0.675
	Stemmed	5	3	0.293	0.196	0.589	0.715
TW3	Unstemmed	8	4	0.273	0.181	0.542	0.673
	Stemmed	3	3	0.297	0.193	0.600	0.710
TW4	Unstemmed	5	4	0.277	0.180	0.551	0.663
	Stemmed	5	3	0.292	0.192	0.585	0.701
TW5	Unstemmed	10	4	0.283	0.185	0.548	0.680
	Stemmed	4	4	0.300	0.190	0.595	0.691
TW6	Unstemmed	10	4	0.283	0.180	0.546	0.671
	Stemmed	5	3	0.294	0.192	0.580	0.700
TW7	Unstemmed	8	5	0.275	0.182	0.543	0.663
	Stemmed	6	4	0.293	0.187	0.575	0.677

Table 5: Effectiveness results for stemmed and unstemmed databases

7 Retrieval Effectiveness Results

In this study, the effects of our stemming algorithm on retrieval effectiveness of Turkish text retrieval are analyzed. The effectiveness measures used in this study are the total number of queries with no relevant documents, Q; the average precision for all queries; and the average recall for all queries. The effectiveness measures are obtained after retrieving 10 and 20 documents, which are typical values for d_s [1]. An effective system yields lower Q and higher precision and recall.

Table 5 shows the effectiveness figures for both the stemmed and unstemmed databases. The experimental results indicate that the matching function has a major part in retrieval effectiveness. The increase in precision due to stemming varies from 0.0 percent ($d_s = 10$ of TW1) to 8.8 percent ($d_s = 10$ of TW3). The lowest values for precision and recall with both databases are obtained with TW1, so we can conclude that cosine function is not a good choice as a matching function.

Table 5 proves that stemming increases the retrieval effectiveness, because lower Q values and higher precision and recall values are obtained with the stemmed database than with the unstemmed database. The only exception is the decrease in precision for $d_s = 10$ of TW1. Excluding TW1, for all the remaining cases, the average percentage increase in precision is 7.05 percent; in recall 5.78 percent. TW3 with $d_s = 10$ gives the highest precision and recall increases, and highest Q decrease.

Although precision values seem too low in all cases, in fact this is misleading because the average of possible maximum precision values for all queries is 0.51 for d_s =10 and 0.30 for $d_s = 20$. The average precision is less than one since the number of relevant documents for some queries is less than d_s (10 or 20).

In this paper, a stemming algorithm for the Turkish language is presented. Using this algorithm a database of Turkish news is indexed. The retrieval performance of the stemming algorithm is then measured in terms of its effect on retrieval performance of a best-match text retrieval system. The results show that stemming reduces the index dictionary size about 65 percent and also improves the retrieval effectiveness.

The increase in the retrieval effectiveness due to stemming is low. The reason may be that the stemming algorithm returns more than one stem for a single word in some cases, causing some unrelevant terms to be conflated. The solution is to select one of the potential stems of the word, but without a semantic investigation it is impossible to determine the correct stem. Considering the fact that stemmers of IR need not be linguistically correct, our stemmer can be modified to select one of the possible stems, either the longest or the shortest one. This will decrease the index size and also shorten the document and query vectors significantly, however may result in understemming or overstemming in various cases. A better approach may be the use of a language independent stemming algorithm, such as successor variety algorithm [4] on Turkish text and compare the effectiveness results with our stemming algorithm.

ACKNOWLEDGEMENTS

We are grateful to Dr. Kemal Oflazer who provided us the Turkish news database, and to the graduate students of CS 533 (Information Retrieval Systems) course who helped us to prepare queries and find the relevant documents for each query.

References

[1] Can, F., Özkarahan, E. A., "Concepts and Effectiveness of the Cover-Coefficient-Based Clustering Methodology for Text Databases", *ACM Transactions on Database Systems*, Vol. 15, No. 4, Dec. 1990, pp. 483–517.

[2] Can, F., "On the Efficiency of Best-Match Cluster Searches", *Information Processing and Management* (to appear).

[3] Dawson, J. L., "Suffix Removal and Word Conflation", *ALLC Bulletin*, Vol. 2, No. 3, 1974, pp. 33–46.

[4] Frakes, W. B., "Stemming Algorithms", in *Information Retrieval Data Structures and Algorithms*, edited by William B. Frakes and Ricardo Baeza-Yates, Prentice Hall, Englewood Cliffs, N.J., 1992, pp. 131–160.

[5] Fujii, H., Croft, W. B., "A Comparision of Indexing Techniques for Japanese Text Retrieval", *ACM-SIGIR Conference* (Pittsburgh, PA, USA, June 1993), 1993, pp. 237–246.

[6] Köksal, A., "Tümüyle Özdevimli Deneysel Bir Belge Dizinleme ve Erişim Dizgesi: TÜRDER", Türkiye Bilişim Derneği 3üncü Ulusal Bilişim Kurultayı, Ankara, 1981, pp. 37–44.

[7] Lovins, J. B., "Development of a Stemming Algorithm", *Mechanical Translation and Computational Linguistics*, Vol. 11, No. 1-2, 1968, pp 22–31.

[8] Mason, T., Brown, D., "lex & yacc", edited by Dale Dougherty, O'Reilly & Associates, Inc., USA, May 1990.

[9] Paice, C. D., "Another Stemmer", *ACM SIGIR Forum*, Vol. 24, No. 3, pp. 56–61.

[10] Popovic, M., Willett P., "Processing of Documents and Queries in a Slovene Language Free Text Retrieval System", *Literary and Linguistic Computing*, Vol. 5, No. 2, 1990, pp. 182–190.

[11] Porter, M. F., "An Algorithm for Suffix Stripping", *Program*, Vol. 14, No. 3, 1980, pp. 130–137.

[12] Salton, G., "Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer", Addison-Wesley, Reading, Mass., 1989.

[13] Salton, G., Buckley, C.,"Term-Weighting Approaches in Automatic Text Retrieval", *Information Processing and Management*, Vol. 24, No. 5, 1988, pp. 513–523.

[14] Schreiner, A. T., Friedman, Jr., H. J., "Introduction to Compiler Construction with UNIX", Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

[15] Solak, A., "Design and Implementation of a Spelling Checker for Turkish", Master's Thesis, Bilkent University, 1991.

[16] Solak, A., Oflazer, K., "Parsing Agglutinative Word Structures and its Application to Spelling Checking of Turkish", Proceedings of COLING'92, Nantes, France, 1992, pp. 26–45.

[17] Solak, A., Oflazer, K., "Design and Implementation of a Spelling Checker for Turkish", *Literary and Linguistic Computing*, Vol. 8, No. 3, 1993, pp. 113–130.

[18] van Rijsbergen, C., J., "Information Retrieval", Second Edition, Butterworths, 1979, pp. 18–19.

[19] "Yeni Yazım Kılavuzu (New Writing Guide)", Ninth Edition, TDK, Ankara, 1977.

[20] "Yeni Yazım Kılavuzu (New Writing Guide)", Eleventh Edition, TDK, Ankara, 1981.