A LINGUISTICALLY MOTIVATED INFORMATION RETRIEVAL SYSTEM FOR TURKISH

by

Fatma Canan Pembe

B.S. in Computer Engineering, Boğaziçi University, 2002

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Computer Engineering

Boğaziçi University

2004

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor, Prof. A. C. Cem Say for his support throughout my thesis work. He has always been involved and has suggested me directions in the thesis talks. I am grateful to the tolerance and encouragement he has shown during my work.

I also thank Assist. Prof. Tunga Güngör and Prof. A. Sumru Özsoy for their advice on my work and for participating in my thesis jury.

I would like to thank Prof. Kemal Oflazer and Özlem Çetinoğlu for providing Turkish WordNet, to be used in our experiments. I thank Prof. Kemal Oflazer also for developing the Turkish Morphological Analyzer used in the morphological part of this work.

Special thanks are to the members of the Computer Engineering Department at İstanbul Kültür University. They have been by my side during this work.

I thank my parents, my brothers and other relatives for their support. I also thank my friends. Without the sincere support of these people, I could not have achieved this work.

ABSTRACT

A LINGUISTICALLY MOTIVATED INFORMATION RETRIEVAL SYSTEM FOR TURKISH

Information retrieval (IR) has become an important application in today's computer world because of the great increase in the amount of web-based documents and the widespread use of the Internet. However, the classical "bag of words" approach no longer meets user expectations adequately. In this context, the use of natural language processing (NLP) techniques comes into mind.

In this thesis, we investigate the question of whether NLP techniques can improve the effectiveness of information retrieval in Turkish. We implemented a linguistically motivated information retrieval system, called TURNA (TUrkish information Retrieval engine based on Natural language Analysis). The system uses knowledge of three different levels of natural language processing in document and query processing: morphological, syntactical and lexico-semantical levels.

Different combinations of these NLP techniques are tested on a set of Turkish documents and queries. The results are evaluated in terms of precision and recall. It is shown that natural language processing techniques, especially stemming and the use of syntactical head-modifier pairs, can improve information retrieval effectiveness in Turkish.

ÖZET

TÜRKÇE İÇİN GELİŞTİRİLMİŞ DİLBİLİME DAYALI BİR BİLGİYE ERİŞİM SİSTEMİ

Ağdaki belgelerdeki büyük artış ve İnternet'in yaygın kullanımı nedeniyle, bilgiye erişim günümüzün bilgisayar dünyasında önemli bir uygulama haline gelmiştir. Ancak, klasik "kelime torbası" yaklaşımı, kullanıcı beklentilerini artık yeterince karşılayamamaktadır. Bu bağlamda, doğal dil işleme (DDİ) tekniklerinin kullanımı akla gelmektedir.

Bu tezde, doğal dil işleme tekniklerinin Türkçe'de bilgiye erişim etkinliğini geliştirip geliştiremeyeceği sorununu ele almaktayız. TURNA adında, dilbilime dayalı bir bilgiye erişim sistemi gerçekleştirdik. Sistem, doküman ve bilgi isteklerini işlemede doğal dil işlemenin üç değişik düzeyindeki bilgileri kullanmaktadır: biçimbilim, sözdizim ve kavramsal sözlük düzeyleri.

Bu doğal dil işleme tekniklerinin değişik birleşimleri Türkçe belgeler ve bilgi isteklerinden oluşan bir küme üzerinde sınanmıştır. Sonuçlar, kesinlik ve geriçağırım açısından ele alınmıştır. Doğal dil işleme tekniklerinin, özellikle gövdeleme ve sözdizimsel tamlayan-tamlanan ikililerinin kullanımının, Türkçe bilgiye erişim etkinliğini geliştirebildiği ortaya çıkarılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

D_j	Document j
d_s	Number of best-matching documents
idf	Inverse document frequency
L_i	Level i
$length_j$	Number of words in document j
M	Number of documents
N	Number of terms in the whole document collection
n_i	Number of documents in the collection in which term i occurs
Q_k	Query k
T_i	Recognizer at level I
tf	Term frequency
w_{ij}	Weight for term i in document j

CFG	Context-Free Grammars
CON	Conjunctive
CPU	Central Processing Unit
FSM	Finite State Machine
GEN	Genitive
GHz	Gigahertz
ID	Identification
IR	Information Retrieval
MB	Megabyte
MHz	MegaHertz
NLP	Natural Language Processing
NOM	Nominative
NP	Noun Phrase
PC	Personal Computer
POS	Part-Of-Speech
POSS	Possessive
RAM	Random Access Memory

TURNA	TUrkish information Retrieval engine based on Natural language Analysis
URL	Uniform Resource Locator

1. INTRODUCTION

Information retrieval (IR) [1] is a relatively old field of research, but its importance has increased significantly after the emergence of the Internet and the increase in the number of digital documents. Every day, millions of queries are run by Internet users in the search engines (e.g., in Google [2]) world wide in order to satisfy their information needs in this huge ocean of information.

An information retrieval (IR) system is an information system; i.e., it is used to store items of information to be processed, searched, retrieved, and presented to various user populations [1]. Therefore, information retrieval systems have common concerns with other information systems such as database systems. As in other information systems, an information retrieval system should organize the stored data efficiently and provide effective search procedures in order to satisfy particular user requests.

One difference of information retrieval systems from other information systems is that, IR systems work on bibliographic records and textual data, whereas the others, such as database systems, generally deal with structured data [1]. The documents and user queries in information retrieval systems are usually natural language formulations. This provides the motivation for using natural language processing techniques in order to improve information retrieval.

The aim of this study is to build an information retrieval system which uses natural language processing techniques in order to provide more effective searches in Turkish. Natural language processing techniques can be applied at different levels: Phonological, morphological, lexical, syntactic, semantic and pragmatic. Moreover, these techniques can be applied at any or a combination of different stages in an information retrieval task, including document processing (and indexing), query processing, query matching, and ranking.

We implemented an information retrieval system, called TURNA (TUrkish information Retrieval system based on Natural language Analysis), in order to test the effects of different natural language processing techniques on retrieval performance.

In the morphological level, we make use of the Turkish morphological analyzer developed by Oflazer [3], in order to see the effect of stemming for Turkish information retrieval. Also, the morphological level information obtained is used for a higher level analysis, namely the syntactical level.

In the lexico-semantical level, we use Turkish WordNet [4, 5], being developed at Sabancı University, as part of the BalkaNet Project. The user queries are optionally expanded in order to also include synonyms, hypernyms/hyponyms, and/or holonyms/meronyms of their individual terms, as a way to improve retrieval effectiveness.

In the syntactical level, we implement a finite-state parser for recognizing a subset of Turkish noun phrases which are expected to carry a higher level meaning. The syntactical information is used to extract head-modifier pairs from the noun phrases in order to discover the effect of using syntactical phrases in indexing and retrieval tasks.

We evaluate the effects of these various techniques on a test set of Turkish documents and queries in terms of the two well-known quality measures in information retrieval, namely precision and recall.

1.1. Outline of the Thesis

Chapter 2 gives a literature survey on information retrieval and natural language processing concepts together with some of the related work in this area. In Chapter 3, a brief description of the architecture and implementation of the TURNA system is given. This chapter is followed by more detailed descriptions of the morphological, syntactical, and lexico-semantical level processing of the system in chapters 4, 5, and 6, respectively. The indexing and retrieval strategies of the TURNA system are given in Chapter 7. This is followed by a performance evaluation of the system in Chapter 8. Chapter 9 is a conclusion.

Appendix A includes the list of stop words used by our information retrieval system. Appendix B gives the natural language queries used in the performance evaluation. Appendix C gives information about the implementation environment.

2. LITERATURE SURVEY

We first give a general description of information retrieval. Then, we present how and why natural language processing techniques can be used in information retrieval, together with some of the related work.

2.1. Information Retrieval

Information retrieval (IR) [1] deals with the representation, storage, organization, and accessing of information items. Theoretically, there is no restriction on the type of the information items. Mostly, however, the information items of interest are textual, including newspaper or research articles, reports, and documents of any kind. Such textual items must be analyzed in order to determine their information content and their ability to satisfy particular information needs of the users.

In a library, each incoming information item, e.g. book, journal, etc, is analyzed and appropriate descriptions are given to them (in the form of library catalog indices) in order to facilitate the desire of the users in finding particular items of interest. In [6], the information retrieval task is described as the "digital twin" of the task of a person in a library searching for items about a certain subject. In both a library and a computerized information retrieval system, the user has an information need which must be formulated either as library indices or *query* terms. In response to the information requests, certain materials are suggested (retrieved) by the system.

Every information retrieval system consists of a set of information items (DOCS), a set of requests (REQS), and a mechanism (SIMILAR) to determine which of the information items possibly satisfy the requirements of specific requests, by mapping specific queries to particular items [1]. The relationship between these parts is given in Figure 2.1.

Theoretically, the documents related to particular queries can be obtained by direct comparison. However, in practice, the similarity is not determined directly. Instead, the

documents are first converted to a special form using an indexing language (LANG) [1]. The user queries are also converted to such a representation using the same indexing language, as seen in Figure 2.2. This mapping of information items to the indexing language can be done manually, automatically, or by a combination of these two methods. The mapping of documents is called the indexing process, whereas the mapping of requests represents the query negotiation process [1].

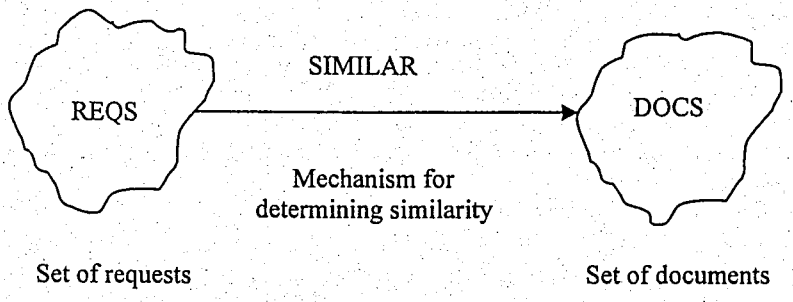


Figure 2.1. A general information retrieval system

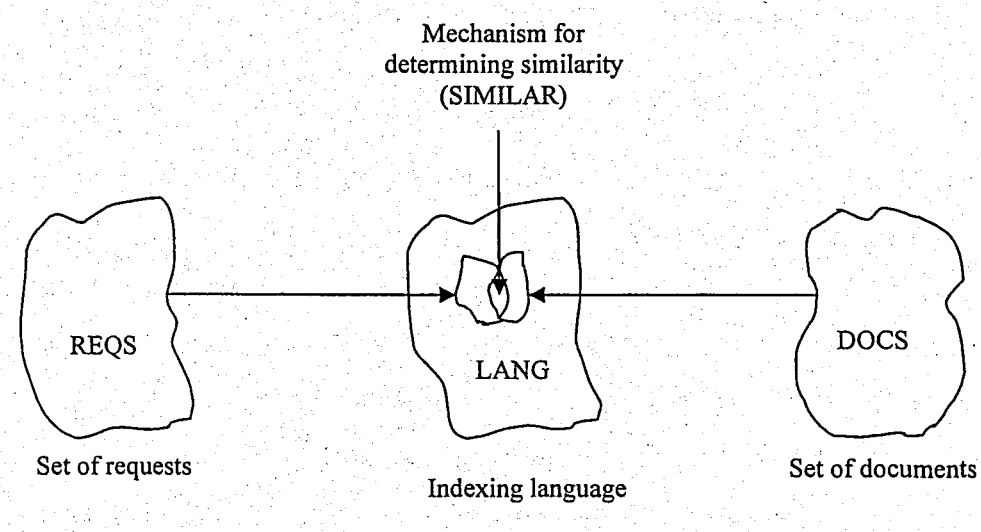


Figure 2.2. Functional overview of information retrieval

The documents which will be retrieved in response to a query are determined using the representations of the documents and the queries based on the indexing language. With

the application of the similarity relation, the documents relevant to a particular query are identified and returned to the user, generally in the order of relevance.

Although IR is a relatively old field with more than four decades of research, its importance has grown tremendously in the last years. The reason is the big increase in the number of documents in digital form and the widespread use of the World Wide Web. People now have access to numerous data thanks to the Internet. However, information retrieval techniques have not developed with the same speed. Usually, the users of current information retrieval systems, such as Google [2] and Yahoo [7], face two important problems. One problem is having to go through hundreds or thousands of irrelevant hits even for a simple search. Another problem is that in such a search, some of the documents will be missing in the results although they are relevant to the user's information need.

In general, web-based IR systems consist of three parts: crawler, indexer and search engine. The *crawler* periodically visits web pages and sends their content to the indexer. The *indexer* maintains a large database in order to keep track of which words are contained in which documents. The task of the *search engine* is, given the user's query and the indexing information, trying to match relevant documents. After this process, the list of retrieved web pages is presented to the user.

There are two metrics commonly used for evaluating the effectiveness of an IR system: precision and recall. *Precision* is about how relevant the search results are, and is defined as the ratio of the number of relevant retrieved documents to the total number of retrieved documents. *Recall* is defined as the ratio of the number of relevant retrieved documents to the total number of relevant documents in the whole document collection. Degraded precision causes a lot of garbage hits, whereas degraded recall corresponds to useful documents missing in the results, both of which are undesirable.

The ultimate goal of every IR system is to increase both precision and recall. The challenge is to understand and represent the documents in the corpus and the user queries in such a way that relevance decisions can be made efficiently.

2.2. Natural Language Processing in Information Retrieval

The documents and user queries under consideration in information retrieval are natural language formulations most of the time. Therefore, available natural language processing techniques can be reviewed, and techniques suitable for information retrieval purposes can be used as a solution to improve the effectiveness of information retrieval in a particular natural language.

In this section, we will first examine the various levels of linguistic methods. After that, the types of linguistic variation, which are responsible for degraded effectiveness in information retrieval systems, will be reviewed. Finally, the motivation for natural language processing in information retrieval will be presented.

2.2.1. Levels of Natural Language Processing

In [1], natural language processing is categorized into six levels: Phonological, morphological, lexical, syntactic, semantic, and pragmatic levels, in the order from the lowest level to the highest.

The *phonological* level is related to the treatment of speech sounds. Therefore, this level is important for speech recognition or speech generation systems, and not used in written text analysis.

The *morphological* level is related to the processing of individual word forms and recognizable portions of words. The prefixes and suffixes can be recognized and removed to form word stems based on the morphological knowledge about a certain natural language.

The *lexical* level is concerned with operations on full words, including common word deletion, dictionary processing of individual words, and expansion of given words using thesaurus information. Some linguistic features, including type information (e.g. noun, adjective, etc), are also recognized in this level, to be further used in syntactical level processing.

The *syntactic* level is designed to group the words of an individual sentence into structural units, such as noun phrases and verb phrases, and to identify subject, verb and object groupings.

The *semantic* level adds contextual knowledge to the information obtained in the syntactic level in order to restructure the text to represent its actual meaning.

Finally, the *pragmatic* level uses knowledge external to the document (e.g., common sense knowledge) in order to improve the interpretation of the text.

All of these levels of natural language processing are related with each other [8]. For example, in the written sentence "I bought Sting's new record", the fact that *record* is used as a noun, and not as a verb, can be understood using lexical and syntactic knowledge. Moreover, owing to semantic and pragmatic levels, we can narrow its actual meaning by concluding that it is a music record, and not a medical record or something else.

The promise of natural language processing is that some of this understanding can be incorporated into a computer system, because natural languages have structure and definable patterns in them. However, it is experienced that, the higher the NLP level in question, the more difficult this incorporation becomes [8]. Fortunately, the full scope of natural language understanding may not be needed in information retrieval applications [1].

2.2.2. Types of Linguistic Variation

Traditional IR systems are based on the simple assumption that if a query and a document have a (key)word in common, then the document is to some extent about the query [6]. This is the well-known "bag of words" approach. That is, each document in the corpus is simply represented as a "bag" of words and tried to be matched with the bag of words representing the user's query. This simple approach does not take the linguistic properties and variation of the text enough into consideration. In [6], linguistic variation is categorized into four groups: Morphological, lexical, syntactical, and semantical variation.

- **Morphological Variation:** For example, *kitap* (book), *kitabım* (my book) and *kitaplar* (books) are three different variants of the same word in Turkish.
- **Lexical Variation:** Different words can represent the same meaning (synonyms). For example, *hikaye* and *öykü* (story) are used interchangeably for the same meaning in Turkish.
- **Syntactical Variation:** The phrases *masanın üstündeki kitap* (the book on top of the table) and *masadaki kitap* (the book on the table) have almost the same meaning.
- **Semantical Variation:** A single word may have different meanings in different contexts. For example, someone searching for *site* in Turkish, can find documents in both web design and real estate contexts.

2.2.3. Motivation for Natural Language Processing in IR

Traditional IR systems usually do not consider the linguistic variation. As a result, they have degraded precision and recall rates. However, the corpus to be searched is essentially a natural language text. When using a search engine, we try to formulate queries in an intelligent way by selecting the right words and the right logical and proximity operators (e.g. AND, OR, NEAR etc) in order to have effective results. The motivation for natural language processing (NLP) in IR is to build some of this knowledge and understanding into the system for better search results [8].

The possible gains which can be expected in using natural language processing techniques in information retrieval context can be stated as follows [1]: First, NLP techniques can make free language formulations possible for the users in entering query statements. This can make the system more efficient and effective by allowing the users to formulate their information requests more precisely and easily.

Second, natural language processing techniques can be used for a more complex analysis and a better content representation of the input documents. As an example, in a traditional IR system, given the query "eş anlamlar sözlüğü" (synonym dictionary), the documents containing all three of the words, i.e. *eş* (same), *anlamlar* (meanings) and *sözlüğü* (dictionary), however in different paragraphs and irrelevantly, can also be wrongly

retrieved. One can try to solve this problem by making an exact phrase search, which would return all the documents containing these three words contiguously and in the order they are entered. However, this time, the documents containing the phrase "eş ve zıt anlamlar sözlüğü" (synonyms and antonyms dictionary) will not be retrieved although they are relevant. To overcome these problems, phrases can be used as indexing terms. To identify the phrases, term co-occurrence statistics and word adjacency operators can be used. However, statistical methods cannot distinguish between cases such as "bilgisayarı derse götürdü" (he took the computer to the lecture) and "bilgisayar dersi" (computer lecture), unlike linguistic methods.

2.3. Related Work

Information retrieval systems can incorporate different levels of NLP. As an example, IRENA [9] is a system which makes use of morphological, lexico-semantical and syntactical analyses of the English language. The work of Solak and Can [10], and the work of Çiftçi [11] are information retrieval systems which use only morphological analyses of the Turkish language. In the following section, these information retrieval systems and a linguistically motivated retrieval system architecture [6] will be overviewed.

2.3.1. IRENA

The IRENA (Information Retrieval Engine based on Natural language Analysis) system consists of four subsystems: a syntactical analyzer, a lexical expander, a morphological expander, and a retrieval system [9].

IRENA accepts queries in the form of English noun phrases. The syntactical analyzer is used for extracting certain keywords from the queries. The keywords are first stemmed and then expanded to obtain all the morphological variants of them. The keywords in the queries are also lexically expanded to obtain all of their synonyms using a thesaurus.

The retrieval subsystem takes this expanded query as input, and selects the documents of the collection containing the terms in the expanded query, by serially scanning each document. Only the fragments of the selected documents containing the

keywords are processed by the syntactical analyzer, and the syntactical relations between keywords are considered. The documents are returned to the user in a ranked fashion based on the observed morphological, lexical and syntactical relations.

2.3.2. Related Work on Stemming in Turkish IR

In [10], the effect of using stemming is evaluated for Turkish text retrieval. The stemming algorithm used is based on a morphological analyzer developed previously by Solak and used for spelling checking of Turkish. During morphological analysis, a dictionary of Turkish root words is used together with rules for Turkish morphophonemics and morphotactics. In this work, all the possible stems of the words are added to the index database.

The system is tested on a corpus of 533 news documents. A stop word list of 293 stemmed terms is used. The vector-space model is used for the documents and queries. The results show that stemming decreases the size of the index database to one third. Different weighting methods are tested for the query and document terms with best-match retrieval; i.e., the best-matching d_s documents are retrieved out of m documents.

The effectiveness measures used in the study are the total number of queries with no relevant retrieved documents, the average precision and recall for all queries. The effectiveness measures are obtained after retrieving 10 and 20 documents which are stated as typical values for d_s. The experiment results indicate that the weighting function has a significant effect on the retrieval performance. The increase in precision due to stemming varies from 0.0 to 8.8 per cent. Precision values are low for all cases which is stated as misleading. The reason is that the total number of relevant documents for some queries is less than d_s (10 or 20).

In this work, the increase in retrieval effectiveness due to stemming is low. The estimated reason is that, the stemming algorithm returns more than one stem in some cases, resulting in irrelevant terms. One approach is selecting only one of the potential stems, but without a semantic investigation, incorrect results may be obtained. Another approach suggested is the use of a language independent stemming algorithm.

Another work on the effectiveness of stemming for Turkish IR is done by Çiftçi [11]. The application built is a multimedia search engine for content-based retrieval of images and text. The stemming technique is used in the textual part of the retrieval system. The stemmer uses the lexicon files from the work of Güngör [12]. One file contains the information on the suffixes and the Finite State Machine (FSM) used. Two different files are used for root words and for proper names in Turkish. Two index files are created for faster access of these two files. The success rate of the stemmer is stated as 92 per cent.

The query server of the application supports two different search methods for the queries: "all words" and "any of the words". The effect of stemming is evaluated on the processing time and retrieval performance of the queries. For this purpose, a part of a Turkish newspaper site in the Internet is indexed. The corpus includes 748 documents (including image and text documents). Ten queries are used each containing two or three words. The queries are classified into two groups. In one group, the roots found by the stemmer have the same meaning used in the target documents. In the other group, at least one root is incorrectly found by the stemmer.

The results show that, when the stemmer is used, the number of relevant retrieved documents increases or stays the same. When the stemmer finds the correct roots for the query words, the number of irrelevant documents returned does not increase much. If the stemmer finds the wrong root for only one of the words, this number again does not increase much. However, if incorrect roots are found for more than one word, the number of irrelevant documents retrieved increases very much.

In Çiftçi's work, the stemming process does not take much time. However, stemming increases the query time, because it increases the number of words that match a query text and the database query time is the most important parameter in the total query time.

2.3.3. A Linguistically Motivated Retrieval System Architecture

The linguistically motivated retrieval system architecture proposed in [6] consists of the following processing steps:

- **Tokenization:** The sentence boundaries and the words within the sentences are detected by considering capitalizations and spacing.
- **Part-of-speech tagging:** A part-of-speech label is assigned to each word; e.g. verb, noun, adjective etc.
- **Morphological normalization:** The goal of morphological normalization is to map different morphological variants of the same word onto one representative form. This is usually done by stemming each word.
- **Collocation identification:** This can be done either by using static collocation lists or word co-occurrence statistics. After collocations are identified, they are treated as a single unit subsequently. For example, "social security" is a collocation.
- **Lexico-semantical normalization:** The problem of lexical variation can be overcome by expanding each word with its synonyms and other related words.
- **Syntactical analysis:** This is generally done for the identification of phrases in the sentences. Given the part-of-speech information for a text, sequences of part-of-speech labels can be examined to identify phrases. For example, the combination of adjective and noun surrounded by other part-of-speech labels is a noun phrase. There may be ambiguous structures. For this purpose, frequency statistics and other heuristics can be used.
- **Syntactical normalization:** The aim is to map different formulations of similar phrases onto one representative phrase form. In a noun phrase, this can be done by identifying and regularizing head-modifier relations.
- **Weighting (and ranking):** Weights are given to the indexing terms, and the retrieved documents are ranked accordingly.

3. THE TURNA SYSTEM

In this chapter, first, the approach and the architecture of the TURNA system is overviewed. Then, the user interface and the implementation of the system is presented. A detailed description of the most important system modules will be given in the following chapters.

3.1. The Approach

The TURNA system operates on three different levels of natural language processing. These are the morphological, lexico-semantical, and syntactical levels.

In the morphological level, the stemming technique is performed. The words are stripped of their suffixes to find their stems. As an example, all the different variations of *okul* (school) found in the documents, such as *okulda* (in the school), *okula* (to the school), *okulum* (my school), will be indexed as *okul* (school). Similarly, the query terms are also stemmed. In this way, documents containing different morphological forms of the same word can be retrieved, improving the recall. This approach may be especially beneficial for languages like Turkish, which is highly rich in suffix usage.

In the lexico-semantical level, we implement the query expansion technique. That is, a given word is expanded to all of its synonyms, and optionally to its hypernyms and hyponyms (the *is-a* relation), holonyms and meronyms (the *part-of* relation). For example, when the user enters a query containing the word *hayat* (life), the query is expanded in order to include also its synonym *yaşam*. With this approach, the documents containing the synonym but not the actual word can also be retrieved, increasing the recall.

In the syntactical level, we considered the "Phrase Retrieval Hypothesis" stated in [6], in order to overcome the shortcomings of the traditional "bag of words" approach. The Phrase Retrieval Hypothesis states that if a query and a document have a phrase in common, then the document is to some extent about the query. This approach encourages the use of phrases instead of just single words as indexing terms. There are several

linguistically motivated IR systems based on the idea of analyzing the text syntactically in order to extract phrases (especially noun phrases) for indexing purposes [13, 14, 15]. The reason for using noun phrases in these systems is that they are important content carriers in almost every natural language [6]. Therefore, we concentrated on Turkish noun phrases in the syntactical part of this work.

The use of noun phrases as indexing terms promises an increase in precision. However, recall would decrease if the problem of syntactical variation is not considered. For example, *nehir kirliliği* (river pollution), *nehirdeki kirlilik* (pollution in the river), and *nehrin yoğun kirliliği* (intense pollution of the river) are different syntactical formulations about almost the same meaning. Therefore, some form of regularization is necessary for phrases. In [6], this kind of regularization is called syntactical normalization. The aim is to map alternative formulations of the same meaning to a single normalized form, such as *nehir+kirlilik*. The normalization technique we use is the extraction of word pairs with head-modifier relation from noun phrases [13, 14, 15]. For example, given the phrase *aile şirketlerinin kurumsallaşması* (institutionalization of family firms), the head-modifier pairs are *aile* (modifier) + *şirket* (head) and *şirket* (modifier) + *kurumsallaşma* (head), which can be easily found because Turkish is a head-final language, i.e., modifiers always precede the head [16]. In this approach, the terms *aile+şirket* and *şirket+kurumsallaşma* can be used as indexing terms. Using such subcompounds in indexing and retrieval instead of whole noun phrases is a widely used solution to the normalization problem.

3.2. The System Architecture

The system (Figure 3.1) contains six distinct modules: Crawler, morphological parser, syntactical parser, indexer, lexico-semantical expander and retrieval engine. In fact, the system provides two different functionalities for two different types of users: the administrator and the ordinary user.

In the administrative part, as in most of the web-based information retrieval systems, the crawler is periodically run (either manually or automatically) in order to access web pages and their content. After this stage, the linguistic processing comes into play. The web documents obtained by the crawler are first morphologically and then syntactically

parsed, and normalized. After these steps, the terms (single words or phrases), which will be used in indexing, are obtained.

In the user part, first, a natural language text query is entered by the user. The query is morphologically and syntactically processed. Then, the obtained terms are lexico-semantically expanded, and submitted to the retrieval engine. Using the index database, the relevant documents are determined, and presented to the user in a ranked fashion.

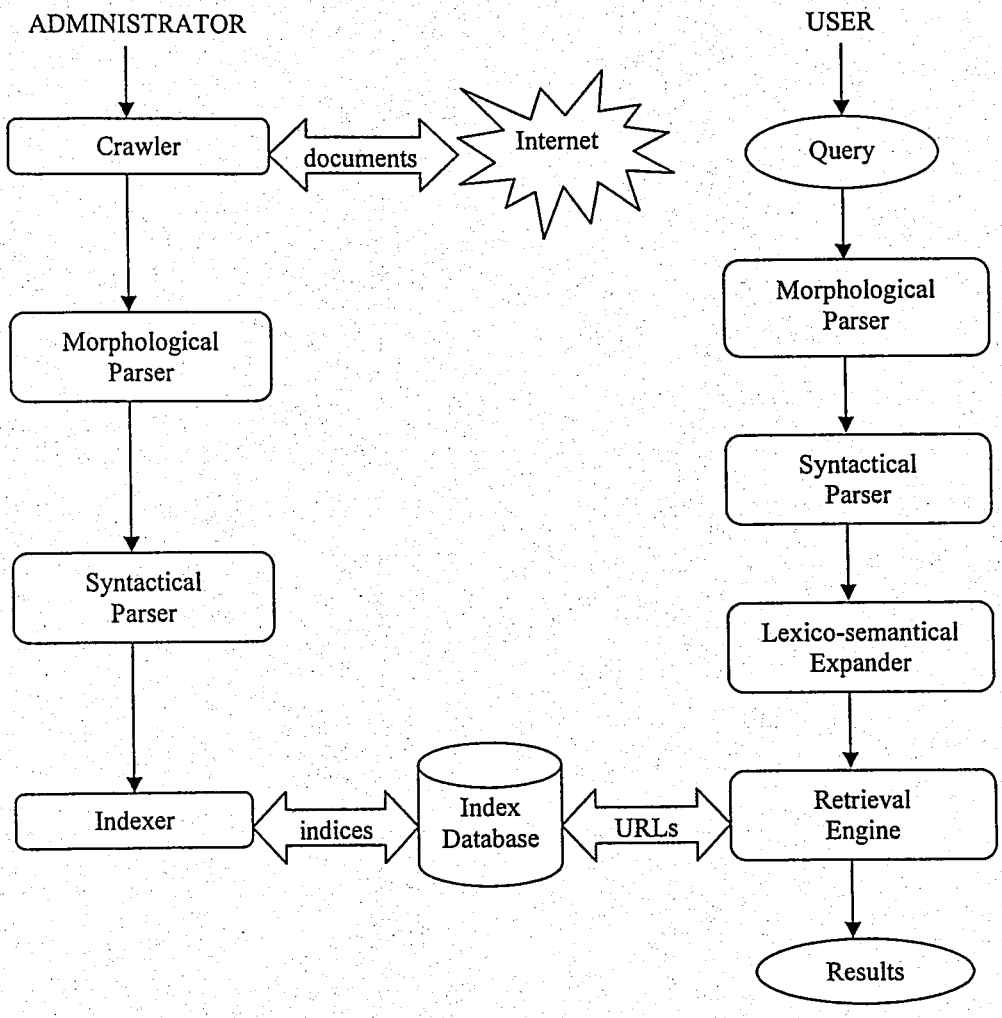


Figure 3.1. Overview of the TURNA System Architecture

3.2.1. Crawler

The crawler is related to the web programming part of the system and is common to almost every web-based information retrieval system. Given the URL (Uniform Resource Locator) of a web site, the crawler retrieves the HTML document residing at this URL and the documents linked by it recursively. The documents retrieved by the crawler are processed in morphological and syntactical levels, and sent to the indexer.

3.2.2. Morphological Parser

The morphological parser used in this work is the Turkish morphological analyzer, developed by Oflazer [3] based on the PC-KIMMO environment [17], and a lexicon of about 23000 root words. The input to the morphological parser is a file containing the words of a given document. The output is the corresponding computed morphological and lexical forms of each word, where each word may have more than one different morphological and lexical parses. The morphological parser serves two different purposes. First, its output is sent to the syntactical parser for a higher level analysis. Second, the stems obtained from morphological parsing are also used as indexing terms by the indexer, as will be explained in Chapter 7. The following is an example output of the morphological parser, i.e., a parse of the word *hizmetini* (the noun "service" with the third person singular possessive and accusative suffixes):

```
hizmet              +sH             +nH
((*CAT* N)(*R* "hizmet") (*POSS* 3SG) (*CASE* ACC) )
```

3.2.3. Syntactical Parser

The syntactical parser accepts the output of the morphological parser as input, that is, the sentences of a document whose words are morphologically parsed. Given this information, the task of the syntactical parser is to extract noun phrases, which will be used in the indexing part. For example, given the sentence *"Bu derslerin en az 12 kredisi Bilgisayar Mühendisliği derslerinden seçilir"* ("At least 12 credits of these courses are selected from Computer Engineering courses"), two possible noun phrases are:

bu derslerin en az 12 kredisi (at least 12 credits of these courses)
Bilgisayar Mühendisliği dersleri (Computer Engineering courses)

3.2.4. Indexer

The noun phrases obtained by the syntactical parser are processed, and head-modifier pairs are extracted from them based on the syntactical relations in the phrases. The head-modifier pairs found are used as indexing terms, in addition to the single words and the stems. The indexer stores all of the indexing terms in a vector space model. The details of the indexing strategy will be given in Chapter 7.

3.2.5. Lexico-semantical Expander

The lexico-semantical expander is the module which is responsible for expanding words in a given user query to all of its synonyms. It is also possible to expand a word to its *hyponyms* and *hypernyms* (the *is-a* relation), and *meronyms* and *holonyms* (the *part-of* relation). For lexico-semantical expansion, we make use of Turkish WordNet, which is being constructed as a part of BalkaNet Project [4, 5].

3.2.6. Retrieval Engine

The retrieval engine takes the user queries which are entered as natural language text. The queries are morphologically and syntactically processed, and lexico-semantically expanded, depending on the choice of the user. Using the indexing information, the query is tried to be matched with the documents in the collection. The list of documents returned by the retrieval system are ranked and displayed to the user.

3.3. The User Interface and Implementation

The interface of the administrative part of the system is implemented in Microsoft Visual Basic 6.0. The crawler is based on the source code available at [18]. In the interface (see Figure 3.2 and Figure 3.3), the base URL address, from which the crawling of web pages will begin, is taken from the user. The user can also specify the number of URLs

kept in memory during crawling, or can give no restriction for that. The functionalities provided by the interface are starting the crawling process, pausing it, viewing the statistics and changing some saving options. The number of web pages scanned is also displayed to the user.

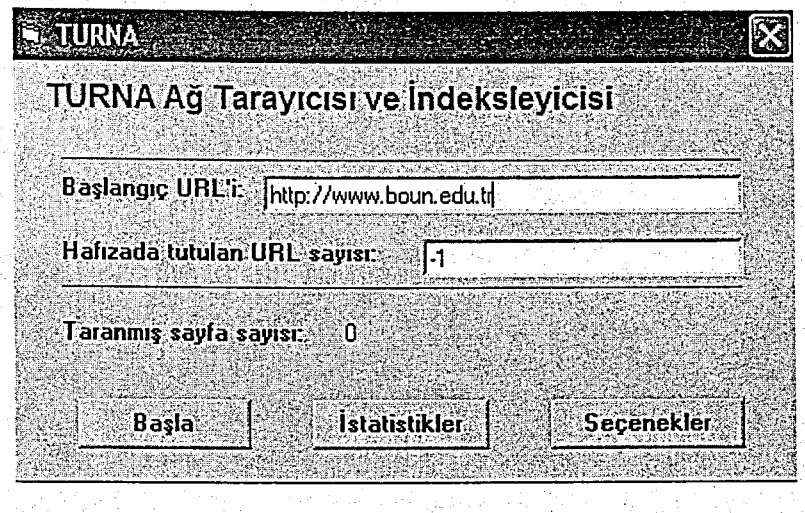


Figure 3.2. The main screen of the Administrative Interface

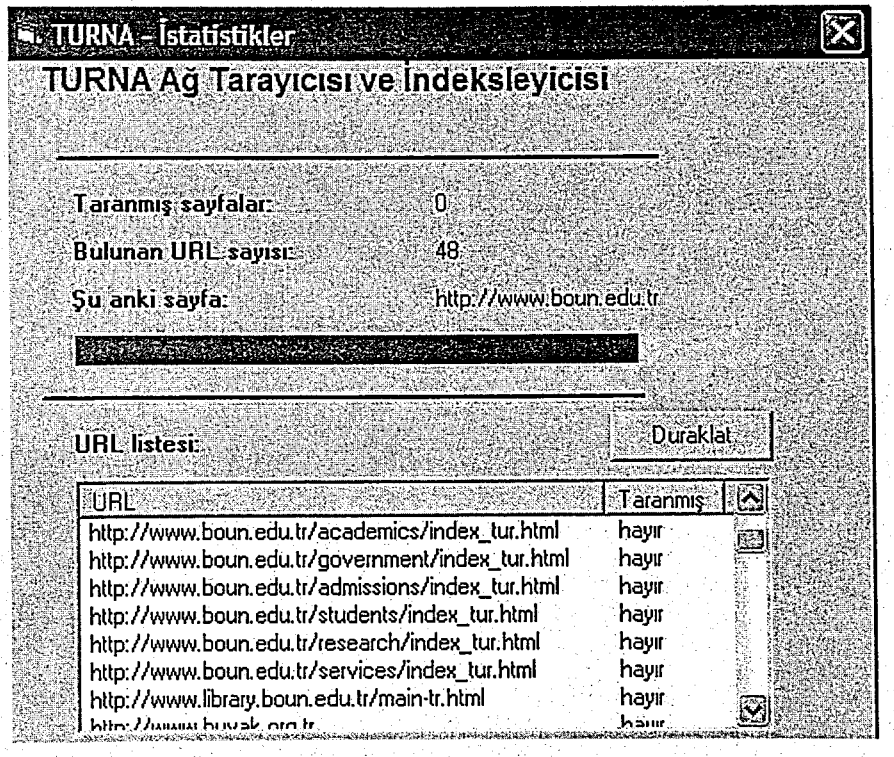


Figure 3.3. The statistics screen of the Administrative Interface

When the administrative interface is loaded, the indexer program starts to run automatically in the background. The indexer program, which is written in C++, analyzes each document crawled by the system, morphologically and syntactically, and builds indices accordingly.

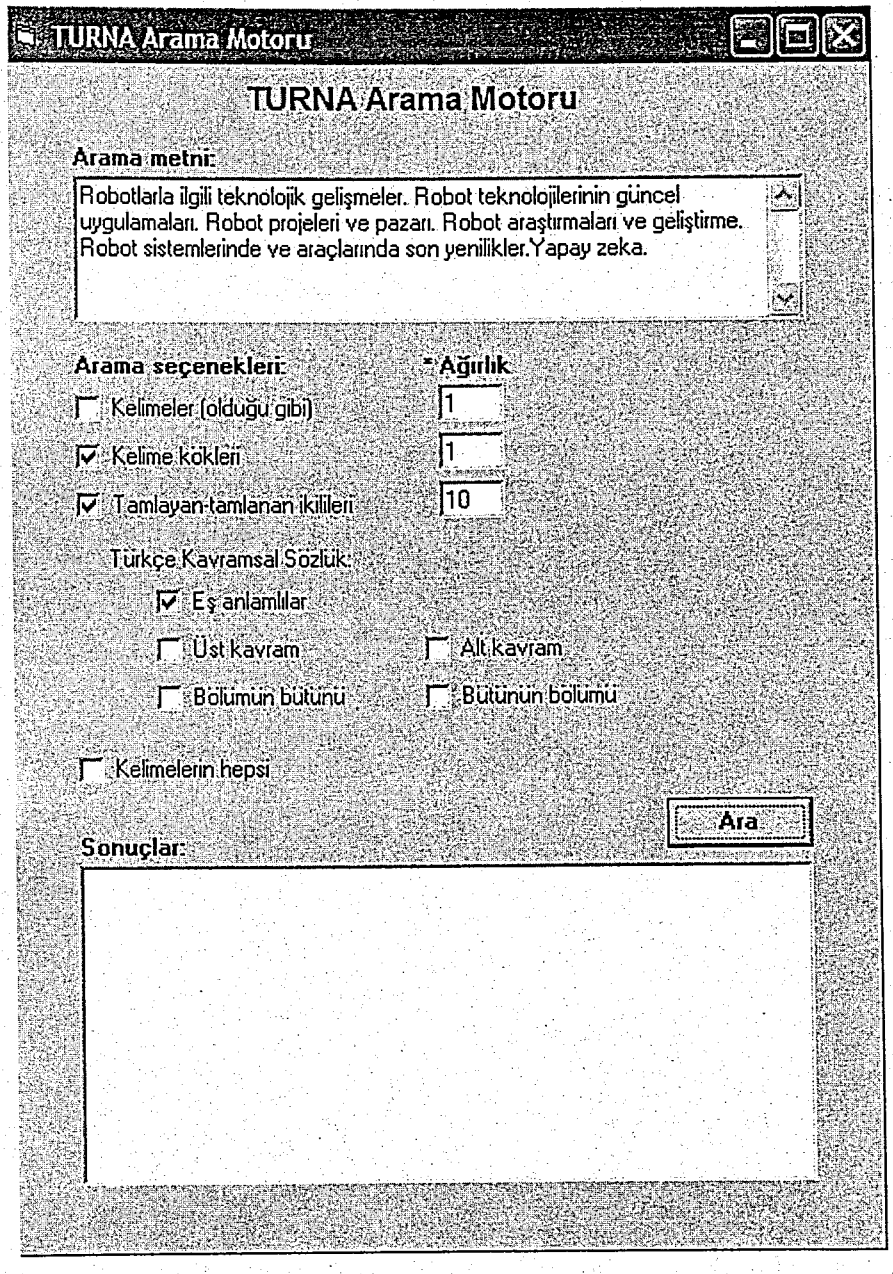


Figure 3.4. The user interface of the system

The interface for ordinary users (Figure 3.4) is also implemented in Microsoft Visual Basic 6.0. In the interface, as in most web search engines, the user enters his query into a

text box. The search options presented to the user include the types and weights of indices used during search (single words, stemmed words and/or head-modifier pairs), use of Turkish WordNet (synonyms, is-a and/or part-of relations), and requiring all of the words in the search.

When the query and the options are submitted, the search program, written in C++, starts to run. The query is morphologically and syntactically processed, and lexically expanded, if desired, by the search program. Using the indexing information, each document in the collection is given a relevance score, and ranked from the most relevant to the least relevant in the user interface. The user can also navigate to the URLs displayed in the output.

4. MORPHOLOGICAL LEVEL

In this chapter, we describe the PC-KIMMO environment and the corresponding Turkish specification, and how they are used in the morphological level of the TURNA project.

4.1. The PC-KIMMO Environment

PC-KIMMO [17] is an implementation of the KIMMO program, named after its inventor Kimmo Koskenniemi, for personal computers. It can be used by computational linguists, descriptive linguists, and natural language processing system developers. The program has two main functionalities: generating (producing) and recognizing (parsing) words using a two-level model of word structure. In this model, a word is represented as a correspondence between its lexical and surface level forms. The lexical form of a word corresponds to its underlying representation, and the surface form corresponds to its realization. For example, in English, the lexical form `spy+s (plural of spy) corresponds to the surface form spies. In the following, ` indicates stress, + indicates a morpheme boundary and 0 indicates a null element [17]:

Lexical representation: ` s p y + 0 s

Surface representation: 0 s p i 0 e s

The PC-KIMMO description of a language is specified by two different files provided by the user, and loaded into the system. One is the *lexicon file*, containing the words, morphemes (the smallest meaningful parts of words) and morphotactic constraints, that is how words are structured from morphemes. The other file is the *rules file*, specifying the alphabet and the phonological rules.

The two functional components of PC-KIMMO are the recognizer and the generator (Figure 4.1). The recognizer accepts a surface form as input, applies the rules and consults the lexicon, and returns the corresponding lexical form with a brief explanation. The generator takes a lexical form as input, applies the rules, and returns the corresponding

surface form. Unlike the recognizer, the generator does not use the lexicon. As stated, the generator produces a word, whereas the recognizer parses a given word.

PC-KIMMO, available for MS-DOS, Microsoft Windows, Macintosh, and Unix, is an interactive program controlled by commands typed at the keyboard. However, it can also be controlled from command line with certain options. The command we are most interested in this project is the *recognize* command. The recognize command takes a surface form from the input, and tries to produce the lexical and morphological forms of it. Both the lexicon and the rules must be loaded before this command can be called. The *file recognize* command is the command used for computing the morphological and lexical forms of words in a given input file.

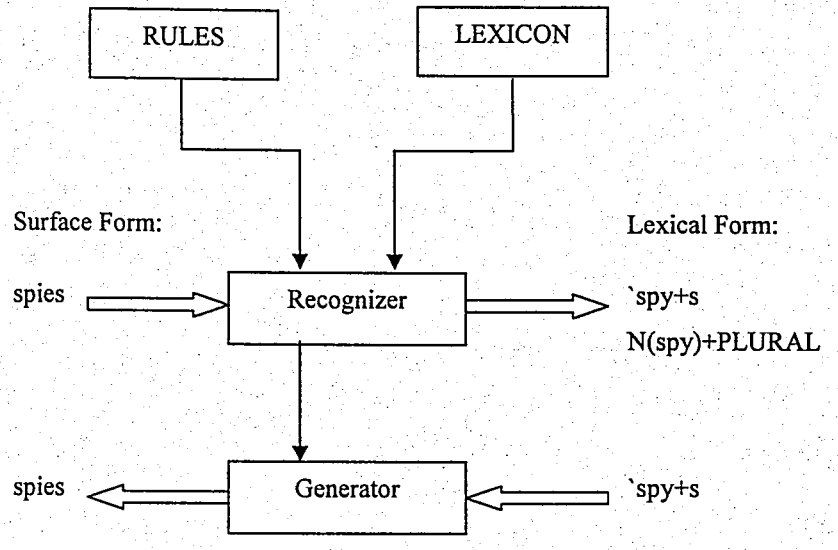


Figure 4.1. The PC-KIMMO system

4.2. Two-level Description of Turkish Morphology

Oflazer has specified a two-level description of Turkish morphology [3], and implemented it for the PC-KIMMO environment. The lexicon contains about 23,000 root words. The implementation includes 22 two-level rules covering Turkish phonetics, and verbal and nominal finite-state machines for the Turkish morphotactics. Almost all the special cases and exceptions of the rules are covered in the implementation.

The phonetic rules constrain the surface realizations of morphological constructions. When a morpheme is affixed to a word, it has to agree with the preceding vowel according to the *vowel harmony* rule. Moreover, there are circumstances when the vowels in the roots and morphemes are deleted. Similarly, consonants in the root words or the suffixes may undergo modifications, and sometimes they are deleted. There are several exceptions to these rules because of the large number of words assimilated from other languages, mostly from Arabic and Persian.

Turkish is an agglutinative language in which derivational and inflectional suffixes are affixed to root words in order to form word structures. An exaggerated but valid example of a Turkish word and its morpheme structure is given in Figure 4.2 where +'s indicate the morpheme boundaries. The meaning of the word can be translated as "as if you were of those whom we might consider not converting into an Ottoman" [3].

OSMANLILAŞTIRAMAYABİLECEKLERİMİZDENMİŞSİNİZCESİNE

OSMAN+LI+LAŞ+TIR+AMA+YABİL+ECEK+LER+İMİZ+DEN+MİŞ+SİNİZ+CESİNE

Figure 4.2. An example Turkish word together with its morpheme decomposition

Turkish has finite-state but rather complex morphotactics [3]. The inflectional suffixes do not change the meaning of the word they are affixed. They only add some features (such as possession or tense) to the functionality of the word. On the other hand, the derivational suffixes change the meaning (and sometimes also the category) of the word they are affixed.

4.3. The Morphological Analyzer

In the morphological level of the TURNA system, the PC-Kimmo system is used together with the lexicon and the rules developed by Oflazer for Turkish. Web documents or queries are submitted to the system as text files, and the *file recognize* command is used.

The morphological parses of the words in a given document are also returned as a text file. We process this file in order to obtain the morphological level information to be used in the syntactical level of the TURNA system.

An example output of the morphological analyzer is given in Figure 4.3. The example contains the four possible parses of the word *etkinlikleri* (meaning "activity" together with some inflectional suffixes) output by the recognizer component.

```
etkinlikleri
0     etkin        +lHk        +lAr        +sH      0
((*CAT* ADJ)(*R* "etkin") (*CONV* N "lik") (*AGR* 3PL)     (*POSS* 3SG) )

0     etkin        +lHk        +lAr        +yH      0
((*CAT* ADJ)(*R* "etkin") (*CONV* N "lik") (*AGR* 3PL)     (*CASE* ACC) )

0     etkin        +lHk        +lArH       0
((*CAT* ADJ)(*R* "etkin") (*CONV* N "lik") (*POSS* 3PL) )

0     etkin        +lHk        +lArH            0
((*CAT* ADJ)(*R* "etkin") (*CONV* N "lik") (*AGR* 3PL)(*POSS* 3PL) )
```

Figure 4.3. An example output of the Turkish morphological parser

The abbreviations we are most interested in the outputs of the recognizer are *CAT* (the category of the root), *R* (the root morpheme), *CONV* (a conversion of the word category due to a derivational suffix), *POSS* (the possessive suffix), and *GEN* (the genitive suffix).

The word categories include noun, adjective, verb, adverb, conjunctive etc. We use this information in the syntactical level of the TURNA system. The original category of the root can be converted to another category as a result of derivational suffix affixation. In the example, the word *etkin* (the word "active" in English), which is originally an adjective, is converted to the word *etkinlik* (activity), which is a noun, as a result of the derivational suffix *lHk*. In the indexing part, we use the form of the word obtained after the addition of

derivational suffixes. In this example, given the word *etkinlikleri*, the word *etkinlik* is used as the indexing term obtained from stemming, and not the word *etkin*.

Inflectional suffixes can be added to nouns or verbs in Turkish. Since we are mostly interested in noun phrases in the TURNA project, we especially considered the inflectional suffixes for nouns in Turkish (Figure 4.4) [12].

The possessive, genitive and case suffixes of a word are added as features to the word in the implementation. This information is used by the syntactical parser in extracting noun phrases.

1. Plural suffix	:	*-lAr*	
2. Possessive suffixes	:	*-(I)m*	*-(I)mIz*
		-(I)n	*-(I)nIz*
		-(s)I	*-lArI*
3. Case suffixes	:	*-DA*	*-CA*
		-DAn	*-lI*
		-(y)A	*-sIz*
		-(y)I	*-(y)lA*
		-(n)InAn	*-(y)IlAn*
Genitive suffix	:	*-(n)In*	
4. Relative suffix	:	*-ki*	

Figure 4.4. Inflectional suffixes for nouns

5. SYNTACTICAL LEVEL

We concentrate on Turkish noun phrases in the syntactical level. After considering two different grammar formalisms, we present the finite-state cascade which we have designed and implemented for Turkish noun phrases.

5.1. Turkish Noun Phrases

In [16], a noun group (noun phrase) is defined as any constituent whose head is a noun. In Turkish, the head noun is the last element of the noun phrase, because Turkish is a head-final language; i.e., modifiers/specifiers always precede the modified/specified. For example, in the noun phrase *öğrencinin kitabı* (the book of the student), the head noun is *kitap* (book) which is specified by the genitive noun *öğrenci* (student).

Noun phrases can have different and important functionalities in a sentence:

- **Subject of a Clause:**
  Çocuklar uyuyor (The children are sleeping)
- **Object of a Clause:**
  Yeni bir kitap aldım (I bought a new book)
- **Complement of a Clause:**
  Çiçeği *evine* götürdü (She took the flower to her house)
- **Object of a Postposition:**
  Alışveriş için geldim (I came for shopping)
- **Specifiers for Possessive:**
  Dönem ödevinin özetini okudu (She read the abstract of the term project)
- **Modifier of a noun group:**
  Bilgisayar mühendisliği bölümünden mezun oldular
  (They graduated from the department of computer engineering)

A noun phrase can be considered as a sequence of segments: the specifier segment, the modifier segment, and the head (see Table 5.1) [16]. Specifiers pick out noun(s) out of

a set of possible nouns, whereas modifiers provide information about the properties of an entity or its relations with other entities.

Table 5.1. Segments of a noun group

Segments	Alternatives	Examples
Specifier	Quantifier	her, bazı, biraz, kimi, herbir, birçok
	Article	bir
	Demonstrative Adjective	bu, şu, o, diğer, ilk, sonuncu
	Genitive noun	bahçenin
	Classifier noun	*mutfak* dolabı
	Relativized noun	evdeki, akşamki
	Relative clause	Postadan çıkan, yolda gördüğüm
Modifier	Quantitative Adjective	dört, yarım, ikişer, üçlü
	Qualitative Adjective	güzel, zor
	Unit noun	bardak, salkım, tane
Head	Common noun	ev, kitap
	Proper noun	Deniz, Ankara
	Pronoun	ben, sen, onlar

The specifier and the modifier segments are optional. Some examples are:

evin *küçük* *mutfağı* (the small kitchen of the house)
specifier + modifier + head

evin *mutfağı* (the kitchen of the house)
specifier + head

küçük *mutfak* (the small kitchen)
modifier + head

mutfak (the kitchen)
head

The order of the specifier and modifier segments is not fixed; e.g., *güzel bir çiçek* (a beautiful flower) and *bir güzel çiçek* (one beautiful flower).

Nouns or noun groups with genitive suffix are used as specifiers within a noun group. They can be used together with other specifiers:

evin bu odası (this room of the house)

When a sequence of genitive nouns is used as specifiers, then each genitive-marked noun is followed by a possessive-marked noun:

evin *mutfağının* *perdesi*
house-GEN kitchen-POSS-GEN curtain-POSS
"the curtain of the kitchen of the house"

Classifier nouns also require a possessive-marked noun group to follow. However, classifier nouns do not take genitive suffix.

mutfak perdesi
kitchen curtain-POSS
"kitchen curtain"

A classifier noun immediately precedes the head noun; i.e., there cannot be other specifiers and modifiers between the classifier noun and the head noun.

eski çocuk odası (correct)
"the old children's room"
çocuk eski odası (wrong)

5.2. Grammar Formalisms

When designing a syntactical analyzer for an information retrieval application, three requirements should be considered [13]. First, the system should be able to process large amounts of text (usually gigabytes) in a reasonable time. Second, the system should be able to process unrestricted text. That is, the system should be robust to the problems of unrestricted natural language corpora, such as unknown words and unrecognized structures. Finally, only a shallow understanding may be sufficient for an IR application

because the goal of an IR system is essentially to classify documents as relevant or irrelevant with respect to a user query. This makes an NLP-based IR application easier to realize than some other NLP applications such as machine translation [13].

Keeping these requirements in mind, we considered two different formalisms for the syntactical processing: context-free grammars and finite-state parsing methods.

5.2.1. Context-Free Grammars

Context-Free Grammars (CFGs) are the most commonly used systems for modeling constituency structure in natural languages [19]. They are also called Phrase-Structure Grammars. A context-free grammar consists of a set of rules (productions) to express how the words of a language can be grouped together to form phrases and sentences. For example, the following is a recursive rule for representing the conjunction of noun phrases to form another noun phrase:

NP → NP CON NP

The symbols used in the productions are divided into two categories: terminal and non-terminal symbols. The right-hand side of the arrow (→) is an ordered list of one or more terminals and non-terminals, whereas the left of the arrow is always a single non-terminal symbol.

Syntactical parsing in context-free grammars can be thought of as a search through the space of all possible parse trees in finding the correct parse of a sentence. There are several methods for parsing with context-free grammars, corresponding to different search strategies, such as top-down, bottom-up and a combination of these methods [19]. However, the problem with context-free grammar parsing is the lack of efficiency and robustness necessary for an information retrieval application.

5.2.2. Finite-State Parsing

Another idea is to use finite-state methods in syntactical parsing. It is stated that finite-state methods are sufficient for applications which do not require complete parses

[19]. Finite-state methods are more efficient than other methods when an unrestricted text corpus is to be parsed, because they rely on simple finite-state automata rather than full parsing [19]. Information retrieval can be considered as such an application.

One inadequacy of finite-state methods is in modeling some types of recursive rules. This is the point where finite-state methods are distinguished from context-free grammars, which rely heavily on recursive rules [19]. Thus, it can be argued that it is not possible to model all of the syntax of a natural language with a finite-state method. However, finite-state methods can approximate the syntax adequately for our purposes. Therefore, we designed and implemented a finite-state parser for the syntactical part.

A finite-state cascade [20] consists of a sequence of finite-state automata. A given sentence is parsed going through this sequence of levels. At each level, certain kinds of phrases are built upon the phrases found in previous levels. In lower levels, "cores" of phrases are identified. These basic phrases are grouped to form more complex phrases in higher levels.

5.3. A Finite-State Parser for Turkish Noun Phrases

After considering previous work done on Turkish grammar [16, 21] and on finite-state modeling of Turkish noun compounds [22], we built a simple finite-state cascade (Figure 5.1) which can recognize a subset of noun phrases in Turkish.

The rules are shown with an arrow ($\rightarrow$) notation for ease of usage, resembling context-free rules. However, these rules are actually compiled together into finite automata and not treated like context-free rules [19].

In this system, parsing corresponds to a series of finite transductions in which spans of input elements are reduced to single elements. The finite-state cascade, which we have designed for Turkish noun phrases, consists of four levels of recognizers (T_i's). These levels of recognizers are hierarchically designed for the recognition of the multi-layered structure in noun phrases.

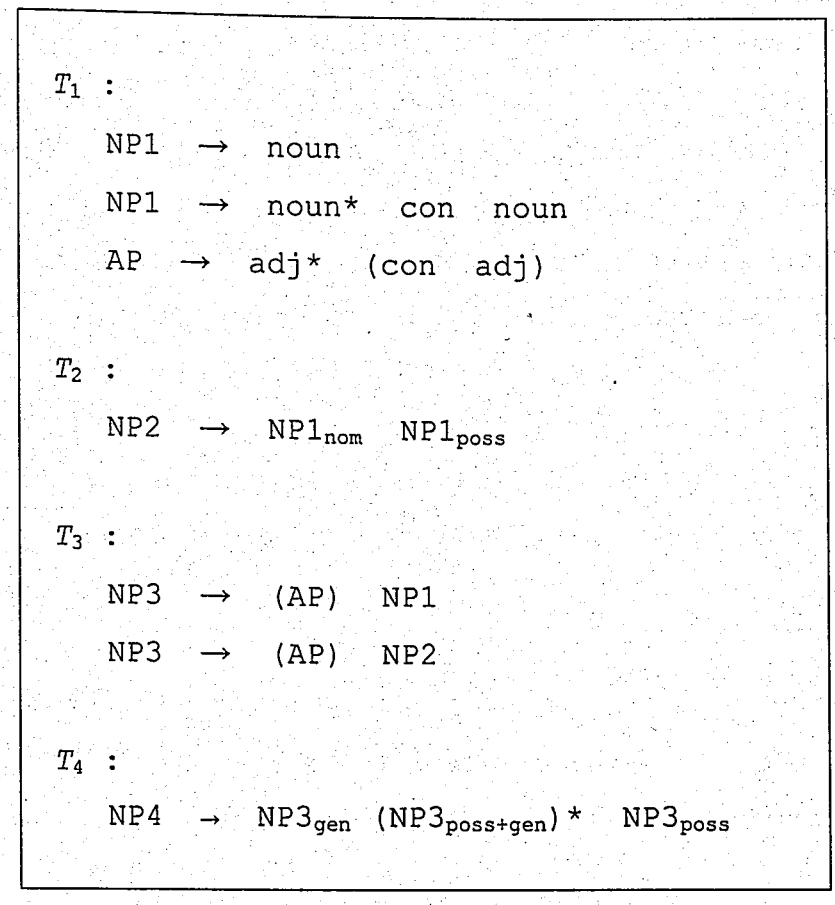


Figure 5.1. A finite-state cascade approximating Turkish noun phrase structure

The first level, T_1, recognizes the basic cores of noun phrases. The first two rules in T_1 are related to the recognition of a single noun, or a sequence of nouns with the last two separated by a conjunctive; e.g. *kitap, kalem ve defter* (a book, a pencil and a notebook). The last rule in T_1 is about recognition of a sequence of adjectives as modifiers or specifiers; as in *küçük ve zeki* (small and intelligent).

The rule in T_2 is for the recognition of the possessive compounds; e.g. *matematik dersi* (the mathematics course). This is the lowest level of modification that can occur within a noun phrase, and nothing may exist between two members of a possessive compound [21].

The rules in T_3 recognize a higher level construction, that is, a noun group preceded by a sequence of adjectives; e.g. *eski masa* (the old table) and *yeni fizik kitabı* (the new physics book).

The rule in the last level, T_4, is for the recognition of genitive-possessive constructions, which is the highest level of modification that can occur between two noun phrases [21]; e.g. *babamın arabası* (the car of my father). This rule can recognize a sequence of genitive-possessive constructions, as in *babamın arabasının tekerleği* (the wheel of the car of my father). This is achieved in one application of the rule; however, the inner structure of the noun phrase, i.e. which possessive noun corresponds to which genitive noun, is preserved.

We omitted the implementation of relative clauses which would require a higher level analysis with subject-object-verb relations; as in *okulda gördüğüm kız* (the girl I saw in the school).

There is ambiguity both in morphological and syntactical level processing, a well-known problem in natural languages. According to our approach, the syntactical analyzer considers all the different morphological parses and produces all the possible different syntactical parses according to the rules.

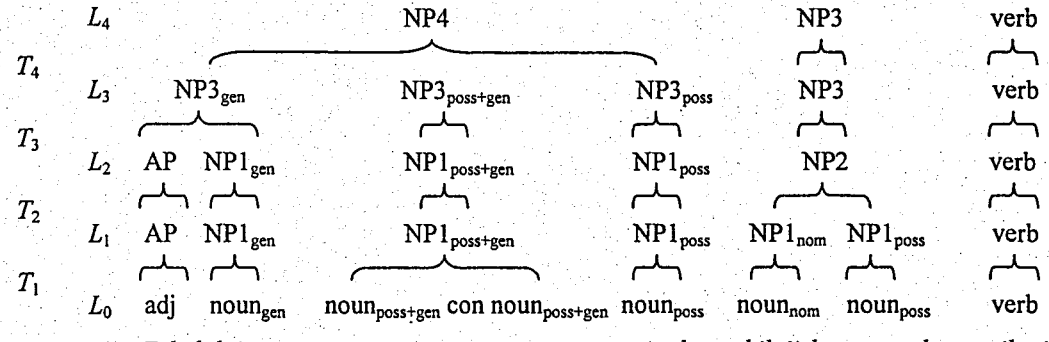


Figure 5.2. Application of the finite-state cascade to an example Turkish sentence

The application of the finite-state cascade on an example sentence is given in Figure 5.2. The syntactical processing starts at level 0 (L_0), given the part-of-speech tags and features of each word in the sentence as input. At each level, L_{i-1} is taken as input and L_i is produced as output, using only the T_i rules corresponding to that level. With each rule application, a sequence of elements is reduced to a single element. If no rule applies at a

level, the element is passed to the higher level without change. At the end of this four-level processing, the noun phrases in the sentence, and their inner structure are obtained.

5.4. Syntactical Normalization

In [6], it is stated that every phrase has a single head according to the linguistic principle of *headedness*. A noun phrase can be represented as a head together with its modifiers in the form of a *phrase frame*:

PF = [*head*, *modifiers*(separated with semicolons)]

The head gives the main concept of the phrase, whereas modifiers are used to make it more precise. Although the head-modifier relation is purely syntactical, it also implies a semantical dependence. This is the motivation for using syntactical head-modifier terms in indexing without the need for a deep semantic analysis [6].

Given the noun phrase *bilgisayar ve matematik dersleri* (computer and mathematics courses), the head is *ders*, and the modifiers are *bilgisayar* and *matematik*. Therefore, the phrase can be represented as:

[*ders*, *bilgisayar*; *matematik*]

In this representation, heads and modifiers can recursively contain other noun phrases as well. The format is:

[[$head_1$, $modifiers_1$], [$head_2$, $modifiers_2$]]

For example, the noun phrase *büyüyen ve gelişen aile şirketlerinin kurumsallaşması* (institutionalization of growing and developing family firms) can be represented as:

[*kurumsallaşma*,
  [*şirket*,
    büyüyen; *gelişen*; *aile*]]

Moreover, the modifier part can be empty, leading to a denotation of the form [*head*]; e.g., [*kitap*].

The structural information obtained from the syntactical analysis is used in extracting the head-modifier pairs from the identified phrases. The parses are processed in different levels to extract head-modifier pairs:

- In level 2, possessive compounds are processed. As an example, the head-modifier pairs *matematik dersi* (the mathematics course) and *bilgisayar dersi* (the computer course) are extracted from the phrase *matematik ve bilgisayar dersi* (the mathematics and computer courses).
- In level 3, noun groups preceded by a sequence of adjectives; are considered. Given the phrase *yeni matematik kitabı* (the new mathematics book), the head-modifier pair *yeni kitap* (the new book) can be extracted. Moreover, the sub-phrases *küçük kalem* (small pencil) ve *kırmızı kalem* (red pencil) can be extracted from the phrase *küçük ve kırmızı kalem* (small and red pencil).
- Finally, in level 4, genitive-possessive constructions are considered. For example, from the phrase *bilgisayarımın eski klavyesinin markası* (the brand of the old keyboard of my computer), the head-modifier pairs *bilgisayar* + *klavye* (computer + keyboard) and *klavye* + *marka* (keyboard + brand) can be extracted in this level.

In the example sentence in Figure 5.2, the head-modifier pairs *eski* + *bilgisayar* (old + computer), *bilgisayar* + *yazıcı* (computer + printer), *bilgisayar* + *tarayıcı* (computer + scanner), *yazıcı* + *sorun* (printer + problem) and *tarayıcı* + *sorun* (scanner + problem) can be obtained from the noun phrase *eski bilgisayarın yazıcısının ve tarayıcısının sorunlarını* (the problems of the printer and the scanner of the old computer) using such a processing on the inner structure of the phrase. The head-modifier pairs obtained in this way are used as indexing terms in addition to words and stems obtained from the documents. The precision of the syntactical parser is 0.70; out of 1357 head-modifier pairs obtained, 954 were correct in a test set of 46 documents.

6. LEXICO-SEMANTICAL LEVEL

In this chapter, we start with an overview of lexico-semantical relations that can take place between words and consider the use of wordnets in natural language processing. Then, we explain how we make use of Turkish WordNet in the TURNA project.

6.1. Lexical Semantics

A lexicon is not merely a simple listing of words and their meanings. It actually has a highly systematic structure about the relations among words and their meanings. *Lexical semantics* is the linguistic study of this meaning-related structure [19]. In this work, we consider the following types of relations between words:

- **Synonymy:** Two words are called synonyms if they can be substituted for one another in some context without changing the meaning or acceptability of a sentence [19]; for example, *siyah* and *kara* (black).

- **Hyponymy and Hypernymy:** These terms correspond to the *is-a* relation between two words [6]. In this type of relation, the more specific word is a *hyponym* of the more general one, whereas the more general one is called a *hypernym* of the more specific one. As an example, *otomobil* (car) is a hyponym of the word *taşıt* (vehicle), and *taşıt* is a hypernym of *otomobil*.

- **Meronymy and Holonymy:** These terms correspond to the *part-of* relation between two words [6]. For example, *tekerlek* (wheel) is a meronym of *otomobil* (car); that is, *tekerlek* is a part of *otomobil*. Also, *otomobil* is a holonym of *tekerlek*; that is, *otomobil* has the part *tekerlek*.

6.2. Use of WordNet

The usefulness of lexical relations in linguistic and computational research has resulted in efforts to create large electronic lexical databases for such relations. WordNet [23] is a well-known and widely used lexical database for English.

The success of English WordNet motivated the development of wordnets for other languages [5]. Wordnets are lexical semantic networks that are based on the notion of synsets. A *synset* is a set of lexical items which are synonymous in a certain context. The following is an example synset from WordNet:

{*person, individual, someone, somebody, mortal, human, soul*}

Semantic relations, such as hyponymy and meronymy, are represented as links between synsets. Some of the noun relations in WordNet are given in Table 6.1 together with examples [19].

Table 6.1. Noun relations in WordNet

Relation	**Definition**	**Example**
Hypernym	From concepts to superordinates	*breakfast → meal*
Hyponym	From concepts to subtypes	*meal → lunch*
Has-Member	From groups to their members	*faculty → professor*
Member-Of	From members to their groups	*copilot → crew*
Has-Part	From wholes to parts	*table → leg*
Part-Of	From parts to wholes	*course → meal*
Antonym	Opposites	*leader → follower*

Turkish WordNet [4, 5] is being developed as a part of the BalkaNet project, which aims to construct a multilingual lexical database for the Balkan languages Turkish, Greek, Bulgarian, Czech, Romanian, and Serbian together with individual wordnets.

In the TURNA project, we make use of the Turkish WordNet, which is still under development at Sabancı University. We have been provided with the XML file containing the Turkish WordNet. Each line of the file contains a synset and the unique ID of that synset. Also the IDs of the synsets with which this synset is in a lexico-semantical relation (such as hypernymy) are given, corresponding to links between synsets. An example portion from Turkish WordNet is given in Figure 6.1. In this example, two different synsets are given. The first synset (with ID ENG20-00006026-n) has two hypernym

synsets. One of its hypernyms (the one with ID ENG20-00003226-n) is also given in the figure.

```
...
<SYNSET>
<ID>ENG20-00006026-n</ID>
<POS>n</POS>
<SYNONYM><LITERAL>insan<SENSE>2</SENSE></LITERAL><LITERAL>kişi<SENSE>1</S
ENSE></LITERAL><LITERAL>şahıs<SENSE>1</SENSE></LITERAL><LITERAL>birey<SEN
SE>5</SENSE></LITERAL><LITERAL>kimse<SENSE>1</SENSE></LITERAL></SYNONYM>
<ILR><TYPE>hypernym</TYPE>ENG20-00003226-n</ILR>
<ILR><TYPE>hypernym</TYPE>ENG20-00005598-n</ILR>
<ILR><TYPE>holo_member</TYPE>ENG20-07463651-n</ILR>
<DEF>Herhangi bir kişi, kim olduğu bilinmeyen kişi</DEF><BCS>1</BCS>
</SYNSET>
...
...
...
<SYNSET>
<ID>ENG20-00003226-n</ID>
<POS>n</POS>
<SYNONYM><LITERAL>organizma<SENSE>2</SENSE></LITERAL><LITERAL>canlı<SENSE
>0</SENSE></LITERAL></SYNONYM>
<ILR><TYPE>hypernym</TYPE>ENG20-00003009-n</ILR>
<DEF>Herhangi bir canlı varlık</DEF><BCS>1</BCS>
</SYNSET>
...
```

Figure 6.1. An example portion from Turkish WordNet

We parse this XML file, in order to obtain this information, and load the WordNet into the memory of our program. Turkish WordNet is used in the query processing part of the TURNA system. The queries entered by the user to the search engine are processed morphologically to obtain the stems. Then, the stems are expanded lexico-semantically according to the choice of the user. That is, the synonyms, hyponyms-hypernyms, and/or meronyms-holonyms of the query terms are also added to the query before the retrieval process takes place. With this approach, the documents containing the synonyms (and

optionally the hyponyms, the hypernyms, the meronyms, and the holonyms) but not the actual word can also be retrieved.

The user can select the expansion of query terms for any one or a combination of the following options:

- Synonymy
- Hyponymy
- Hypernymy
- Meronymy
- Holonymy

7. INDEXING AND RETRIEVAL

In this chapter, we first overview the Vector Space Model used in the indexing and retrieval part of the TURNA system. Then, we present the indexing and retrieval strategies of the TURNA system.

7.1. Vector Space Model

The Vector Space Model [19] is a widely used model in information retrieval. In this model, documents and queries are represented as vectors of features which correspond to the terms contained within the whole document collection. The value of each feature in the vectors indicate the presence or absence of a given term in a given document. In the following, D_j and Q_k denote a particular document vector and a particular query vector, respectively, where t's correspond to an ordered list of term values, and N is the total number of terms in the whole document collection.

$$D_j = (t_{1j}, t_{2j}, t_{3j}, \ldots\ldots, t_{Nj})$$
$$Q_k = (t_{1k}, t_{2k}, t_{3k}, \ldots\ldots, t_{Nk}) \qquad (7.1)$$

In the simplest case, the features of the vectors (t's) take the values of one or zero, indicating the presence or absence of a term in a given document or query. In this approach, the relevance of a document to a query can be determined simply by counting the number of the terms common to both the document and the query. The similarity between the document vector D_j and the query vector Q_k can be specified as the following:

$$Sim(Q_k, D_j) = \sum_{i=1}^{N} t_{ik} \times t_{ij} \qquad (7.2)$$

The inadequacy in the use of binary values (0 or 1) is that it does not differentiate between the importance of terms; that is, all the terms are considered equally important. A better solution is to use numerical *weights* instead of just ones and zeroes. In this

representation, terms which are more important in the document or the query will receive higher weights (shown as w's):

$$D_j = (w_{1j}, w_{2j}, w_{3j}, \ldots\ldots, w_{Nj})$$
$$Q_k = (w_{1k}, w_{2k}, w_{3k}, \ldots\ldots, w_{Nk}) \qquad (7.3)$$

The representation of documents as vectors of term weights allows us to view the document collection as a matrix of weights, called a *term-by-document matrix*, where w_{ij} corresponds to the weight of the term i in the document j [19]. In this matrix, the columns represent the documents of the collection and the rows represent the terms. In Table 7.1, a sample document collection and a user query is given. The representation of these documents and the query in a simple (binary) Vector Space Model is given in Table 7.2. Note that stemming is not considered in this simple system, and the terms *system* and *systems* are evaluated as different terms, and as a result, the term *systems* is not represented in the query vector (because there is no such term in the document collection).

Table 7.1. A sample document collection and a query

	Content
Document 1	The Implementation of the TURNA System
Document 2	User Interface of the TURNA Information Retrieval System
Document 3	Implementation of the TURNA Information Retrieval System
Document 4	Introduction to Modern Information Retrieval
Query (Q)	Information retrieval systems

The representation of documents and queries as vectors provides the infrastructure of an information retrieval system. Such a system can accept a user query, create a vector representation for it, compare it with the vector representations of the documents in the collection, rank the resulting list of documents according to their relevance, and present it to the user.

Table 7.2. The representation of the sample documents and the query in the Vector Space Model

Terms	Documents				Q
	1	2	3	4	
the	1	1	1	0	0
implementation	1	0	1	0	0
of	1	1	1	0	0
TURNA	1	1	1	0	0
system	1	1	1	0	0
user	0	1	0	0	0
interface	0	1	0	0	0
information	0	1	1	1	1
retrieval	0	1	1	1	1
introduction	0	0	0	1	0
to	0	0	0	1	0
modern	0	0	0	1	0

The method used to assign term weights in the document and query vectors has a big impact on the effectiveness of an information retrieval system [19]. Two factors are considered as useful in the weight assignment: term frequency within a single document and the distribution of terms across a document collection.

In the TURNA system, the weights are computed using the widely used weighting scheme $tf \times idf$ where *tf* (term frequency) refers to the number of times a particular term occurs in a given document or query, and *idf* (inverse document frequency) is a measure of how rarely a particular term appears across all of the documents in the collection. The *idf* value for each term i, and the weights for term i in document j are computed as follows (where M is the total number of documents in the collection and n_i is the number of documents in which term i occurs):

$$idf_i = \log(M / n_i) \tag{7.4}$$

$$w_{ij} = tf_{ij} \times idf_i \tag{7.5}$$

With this scheme, frequently used terms in a particular document will have higher weights, because they may reflect the meaning of the document more strongly than less

frequently used terms. Moreover, terms common to most of the documents in the collection will receive lower weights and vice versa, because it is thought that terms which are limited to a few documents are useful in distinguishing those documents from the rest of the collection.

In our system, the similarity between a query Q_k and a document D_j is computed using the dot product formula normalized with document length ($length_j$ which is defined as the number of words in document j) in order not to favor longer documents [24]:

$$Sim(Q_k, D_j) = \frac{\sum_{i=1}^{N} w_{ik} \times w_{ij}}{\sqrt{length_j}} \qquad (7.6)$$

7.2. Indexing Strategy

Each time a new Web page (document) is crawled by the TURNA system, it is processed in the morphological and syntactical levels in order to obtain the indexing terms from it. There are three types of indexing terms (with the type identifiers given in parentheses):

- Single words before any processing (0)
- Stems obtained after morphological processing (1)
- Head-modifier pairs obtained after syntactical processing (2)

An index in the TURNA system is a pair consisting of the actual indexing term and its type. In Table 7.3, the indexing terms obtained from the phrase *öğrenci kulüpleri* (student clubs) are given. The single words (type 0), the stemmed words (type 1) and the syntactical head-modifier pairs (type 3) are identified as indexing terms. Note that the indexing terms (OGrenci, 0) and (OGrenci, 1) are identified as different indexing terms; one as the original word in the text and the other as a stem. This information will be used by the retrieval engine in providing the users with options on the type of retrieval strategy he/she would like to use. Also note that, all the characters are in lowercase except for the Turkish

characters *ç, ğ, ı, ö, ş, ü* that are converted to corresponding uppercase characters (*C, G, I, O, S, U*) to avoid any Turkish character problems.

Table 7.3. Indexing terms for the phrase *öğrenci kulüpleri*

Indexing value	**Type**
OGrenci	0
kulUpleri	0
OGrenci	1
kulUp	1
OGrenci+kulUp	2

The document information of the collection is stored in a text file (*documents.txt*) with the following fields for each document, and updated automatically with the addition of new documents:

- a unique document identifier
- the URL of the document
- the length of the document (in terms of word count, which is to be used in normalization of the corresponding document vector)

The indexing terms of the whole document collection are also stored in a text file (*allIndex.idx*). Each time a new document is indexed, this ordered list of the indexing terms of the whole document collection is searched. If the indexing term is not already present in the list of collection's indexing terms, it is added to the list. In this way, the list of the indexing terms of the whole document collection is built incrementally with the addition of new documents to the system. The indexing system also keeps track of the document frequencies of each indexing term; that is, in how many documents a particular indexing term appears. This information is also updated with the addition of new documents to the system. This information is used in determining the weights of the document and query vectors together with the term frequencies in a particular document and query.

When a new document is added to the system, the corresponding document vector is also created at the same time according to the ordered list of the indexing terms of the whole document collection. This is accomplished by incrementing the value of the corresponding vector position (the number of occurrences of a particular indexing term in that document) for each of the indexing terms obtained from that document. Each document vector is stored in a text file (*<docID>.vec*). These data are used in the query matching process by the retrieval engine.

Another technique we use in indexing term selection is utilizing a stop word list. A *stop word list*, which is commonly used in information retrieval systems, is a list of high frequency words and closed-class that are eliminated from document and query representations. One motivation is that high frequency and closed-class terms, such as *ama* (but), *ve* (and), etc. carry little semantic weight when they are isolated from the rest of the sentence and selected as indexing terms. Second, eliminating them can save considerable space in the index files [19]. Stop words are usually selected from the part-of-speech tags including pronouns, postpositions, and conjunctives. The Turkish stop word list we built for the TURNA system is given in Appendix A.

7.3. Retrieval Strategy

The query entered by the user in the search engine is processed in the morphological, syntactical and lexico-semantical levels and converted to a vector representation. However, since queries are usually sparse vectors with zero values everywhere except in a few term positions, we store the queries in a different data structure. Only the query term values that are nonzero are stored and used in the calculations.

The user preferences for a particular search (designated as checkboxes) are taken from the user interface as command line arguments to the search program. The choices of the user determine which type of indexing terms will be used during matching of a particular query to the documents in the collection. Presented with the three options (single words (0), stems (1), and head-modifier pairs (2)), the user can select any or a combination of them. For example, if the user selects the options of stems and head-modifier pairs, then in the retrieval, exactly those terms of the document vectors of the collection (the indexing

terms with the selected types; i.e. 1 and 2 in this case) are used to compute the similarity measure with those terms (with types 1 and 2) of the query. The user can also specify that all of the words in the query should exist in the returned documents.

The choice of the user also determines whether or not query expansion will be performed using the Turkish WordNet, and if it will, then which relations (synonymy, hyponymy etc) will be included.

In accordance with the user preferences, the dot product of the query vector is computed with each document vector in the collection. The resulting nonzero numerical scores are sorted in descending order, and presented to the user with the corresponding document URLs.

8. PERFORMANCE EVALUATION

In this chapter, we evaluate the performance of the TURNA system. For this purpose, we first introduce the quality measures, the corpus and the queries we have used. Then, we give the experiments and present their results.

8.1. Quality Measures

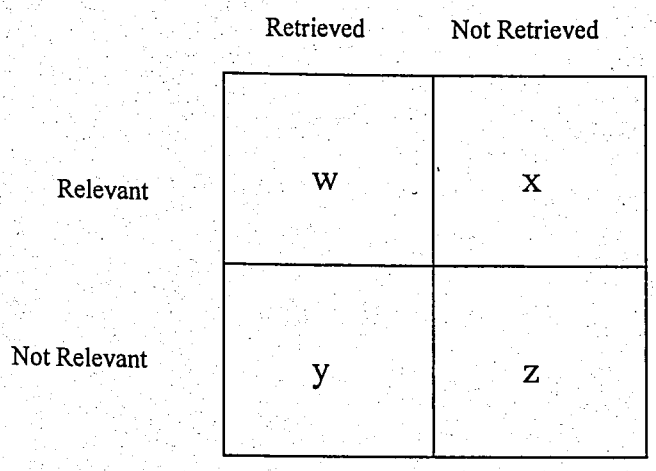
An information retrieval system can be evaluated according to the following quality measures from the user viewpoint [1]:

- **Recall:** The ability of the system to retrieve useful documents.
- **Precision:** The ability to present only the relevant items and filter out useless ones.
- **User effort:** The intellectual and/or physical effort required from the users in formulating the queries, conducting the search and viewing the results.
- **Response time:** The time between the submission of a user query and the presentation of the results by the system.
- **Form of presentation:** The presentation of the outputs which influences the user's ability to utilize the materials.
- **Collection coverage:** The extent to which relevant items are covered by the system.

In the performance evaluation, we will mostly concentrate on precision and recall. A document collection can be divided into four partitions considering the combinations of whether the documents are relevant or not and whether they have been retrieved or not in response to a particular user query (Figure 8.1). In this representation, the precision and recall can be defined as:

$$Precision = w / (w + y) \qquad (8.1)$$

$$Recall = w / (w+x) \tag{8.2}$$



w : relevant and retrieved items
x : relevant and not retrieved items
y : not relevant and retrieved items
z : not relevant and not retrieved items

w + y : retrieved documents
w + x : relevant documents
w + x + y + z : documents in the collection

Figure 8.1. Partititions of a document collection

More formally, precision and recall can be defined as in 8.3 and 8.4. Given the corpus, denoted with the set O, and the parameter λ depending on the retrieval strategy, if a query q retrieves the subset B_q of O when in fact the subset A_q of O was intended, then precision and recall are defined as [9]:

$$Precision_{\lambda}(q) = \frac{|A_q \cap B_{q\lambda}|}{|B_{q\lambda}|} \tag{8.3}$$

$$Recall_{\lambda}(q) = \frac{|A_q \cap B_{q\lambda}|}{|A_q|} \quad (8.4)$$

An alternative measure to precision and recall is F measure [25]. This method changes the precision value p and recall value r into 1-dimensional scalar, and it is in agreement with the harmonic mean of precision and recall:

$$F = 2pr / (p+r) \quad (8.5)$$

8.2. The Corpus and the Queries

TREC [26] is a well-known organization which provides standard document collections, queries, and relevance decisions to be used in IR research experiments for languages such as English and Spanish. However, currently there is no such standard document and query collections for Turkish. Therefore, for the performance evaluation, we have collected Turkish documents about different topics (including human resources management, smoking, advances in robotic technologies, etc.) from the Internet. Some statistics about the corpus are given in Table 8.1. Although the size of the corpus is small with respect to TREC Databases, which contain tens of thousands of documents, it is a reasonable size to start the tests.

Table 8.1. Statistics about the corpus used in the experiments

Number of documents	615
Average number of words per document	231
Number of words in the collection	142,533
Total size	1.07 MB

We formed five descriptive, natural language queries which are given in Appendix B. Each document in the corpus was manually judged as relevant or irrelevant (in a binary fashion) beforehand with respect to each query, as in the TREC test sets. These relevance judgements are used in evaluating the effectiveness of the system.

8.3. Experiment Setup

In order to see the comparative performance of different indexing and retrieval approaches, we used the runs (numbered from 0 to 9) given in Table 8.2. The simplest case, denoted with Run 0, is using only single unprocessed words as indexing terms. This is followed by approaches with different combinations of using stems and/or head-modifier pairs. Another option is the use of lexico-semantical expansion in the query terms.

Table 8.2. Different combinations of indexing and retrieval approaches used in the experiments

Indexing and retrieval approach	**Runs**									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Single words	+									
Stemmings		+	+	+	+	+	+	+	+	+
Head-modifier pairs					+	+	+			
Head-modifier pairs heavily weighted								+	+	+
Synonym			+	+		+	+		+	+
Hypernym/Hyponym				+			+			+
Holonym/Meronym				+			+			+

In Run 1, only stems are used as indexing and query terms. If a stem is not found for a given word, the word is indexed as it is. In Run 2, in addition to the use of stems, query terms are also expanded with their synonyms. In Run 3, the query terms are also expanded with their hypernyms, hyponyms, holonyms and meronyms in addition to their synonyms.

Runs 4, 5 and 6 are obtained by using syntactical head-modifier pairs in addition to the stems used in runs 1, 2 and 3, respectively. The difference of runs 7, 8 and 9 from runs 4, 5 and 6 is that, in these runs syntactical head-modifier pairs are ten times more heavily weighted during retrieval.

8.4. Results

The performance results are obtained on a personal computer whose system feauters are given in Appendix C. The average processing times of different stages in document indexing and query processing for the test set are given in Table 8.3. In the document indexing part, the morphological processing stage includes the time between the input of a document to the Morphological Analyzer and its computed morphological analysis. In query processing, this is the time for the processing of the query by the Morphological Analyzer. As seen, the morphological processing is about the same for both the documents and the queries.

Table 8.3. Average time durations of different processing stages in the TURNA system

Stage	**Indexing**	**Query Processing**
Morphological processing	2.78 sec / document	2.75 sec / query
Syntactical processing	13.52 sec / document	0.06 sec / query
Lexico-semantical expansion	-	5.37 sec / query
Matching	-	189.37 sec / query
Total	16.30 sec / document	198.39 sec / query

The syntactical processing is more dependent on the size and the complexity of the text. As a result, the syntactical processing of the documents takes much more time than that of queries. Moreover, the finite-state parsing method used resulted in reasonable processing times.

The lexico-semantical expansion technique applies only to the queries, and not to the documents. The time for expanding a query lexico-semantically is quite long. The reason for this is that we load the WordNet from an XML file into the memory as a list of lexical records, and currently use sequential search. Usage of more efficient data structure and algorithms for the wordnet would decrease this time, which is also one of the aims of Turkish WordNet project.

The total processing time of a query includes the times of its morphological and syntactical processing, its lexico-semantical expansion, sequential matching process with each document vector in the collection, and ranking the results. This time is quite long; however, it is mostly dominated by the sequential query matching process as seen in Table 8.3. In the TURNA system, the query vector is tried to be matched by loading each document vector to the memory from the corresponding files and computing the dot product. The reason for such a long time is due to the big number of file operations in that stage. One solution may be keeping all the document vectors in the memory; however this would increase the memory usage very much depending on the number of documents in the collection. Since this processing time is independent of the natural language processing techniques involved and we are working on an experimental system currently, we have not much concentrated on that aspect.

We compared the processing power of the machine on which we made the tests with the machines of Google [27]. Google uses clusters of more than 15,000 inexpensive PCs instead of fewer but more expensive servers. The servers of Google range from single-processor 533-MHz Intel-Celeron-based servers to dual 1.4-GHz Intel Pentium III servers. Google's search application is suitable to extensive parallelization. The overall index is partitioned which enables a single query use multiple processors improving the performance.

The number of indexing terms belonging to different index sets is given in Table 8.4. As seen, the number of indexing terms is reduced to about one half when stems are used as indexing terms instead of unprocessed words.

Table 8.4. Number of different types of indexing terms in the experiment

Index type	**Number of indexing terms**
Single words	28,152
Stemmed words	13,185
Head-modifier pairs	7,571
Total	48,908

For each run, the recall value is fixed at certain levels, to be able to compare the precision values obtained [1]. Run 0 represents the traditional approach in which only single words are indexed without the use of morphological, lexico-semantical and syntactical analyses. As seen, the incorporation of morphological information in Run 1, by means of stemming, improves the effectiveness significantly. The introduction of lexico-semantical expansion in runs 2 and 3 resulted in a decrease in average precision (Table 8.5).

In a related work on the effect of stemming for Turkish text retrieval [10], precision and recall are computed for the first 10 and 20 documents in the ranked results that are returned. The number of relevant documents in that collection is less than 10 and 20 for some of the queries, which can result in small and misleading numbers for precision. In our experiment, we show precision values at different recall levels which is more precise.

Table 8.5. Interpolated precision results of runs 0, 1, 2 and 3

Recall	Run 0	Run 1	Run 2	Run 3
0.0	0.32	0.83	0.70	0.67
0.1	0.32	0.83	0.70	0.67
0.2	0.35	0.73	0.67	0.64
0.3	0.37	0.74	0.67	0.62
0.4	0.38	0.68	0.58	0.54
0.5	0.38	0.66	0.57	0.51
0.6	0.32	0.64	0.57	0.51
0.7	0.30	0.56	0.55	0.43
0.8	0.25	0.53	0.41	0.36
0.9	0.20	0.39	0.32	0.23
1.0	0.12	0.25	0.17	0.13

The introduction of syntactical information, with head-modifier pairs as indexing terms, in runs 4, 5 and 6, generally gives improved results (Table 8.6). However, this increase is much better when head-modifier pairs are more heavily weighted during retrieval as in runs 7, 8 and 9 (Table 8.7).

Table 8.6. Interpolated precision results of runs 4, 5 and 6

Recall	Run 4	Run 5	Run 6
0.0	0.80	0.81	0.68
0.1	0.80	0.81	0.68
0.2	0.78	0.67	0.65
0.3	0.76	0.67	0.65
0.4	0.71	0.62	0.56
0.5	0.66	0.58	0.51
0.6	0.66	0.58	0.51
0.7	0.58	0.54	0.42
0.8	0.52	0.41	0.37
0.9	0.38	0.33	0.24
1.0	0.24	0.16	0.13

Table 8.7. Interpolated precision results of runs 7, 8 and 9

Recall	Run 7	Run 8	Run 9
0.0	0.93	0.87	0.79
0.1	0.93	0.87	0.79
0.2	0.78	0.76	0.69
0.3	0.76	0.71	0.65
0.4	0.72	0.64	0.62
0.5	0.72	0.58	0.54
0.6	0.67	0.58	0.52
0.7	0.61	0.54	0.46
0.8	0.52	0.44	0.38
0.9	0.39	0.33	0.24
1.0	0.26	0.17	0.13

The precision-recall graph for runs 0, 1, 4 and 7 are given in Figure 8.2. As seen, the linguistic processing in runs 1, 4 and 7 gives a great improvement over the traditional approach with no linguistic processing; i.e. Run 0. The best results are obtained in Run 7, where stems and heavily weighted head-modifier pairs are used in indexing and retrieval.

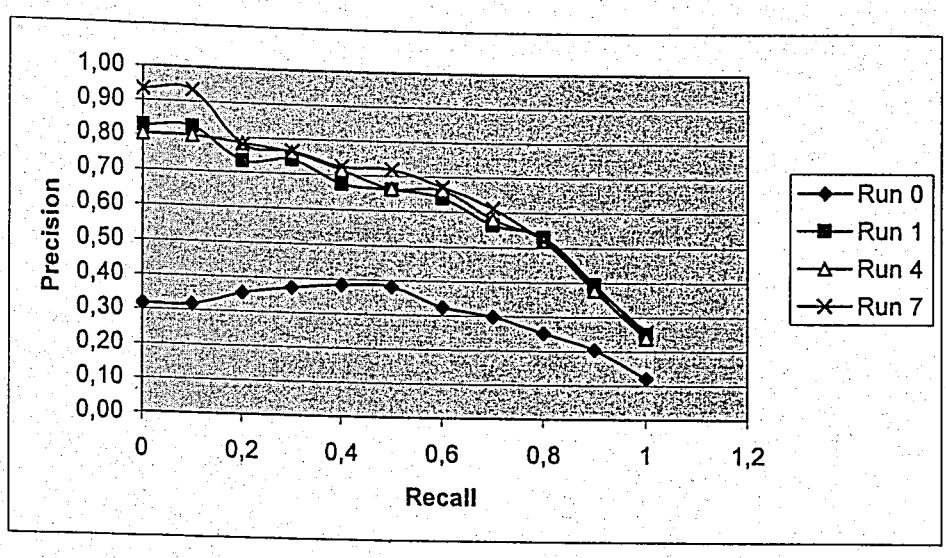


Figure 8.2. Precision-Recall graph for runs 0, 1, 4 and 7

The precision-recall graph for runs 7, 8 and 9 are given in Figure 8.3. These are the runs with the best results in general. However, again, the use of lexico-semantical expansion in runs 8 and 9, resulted in a worse performance than Run 7.

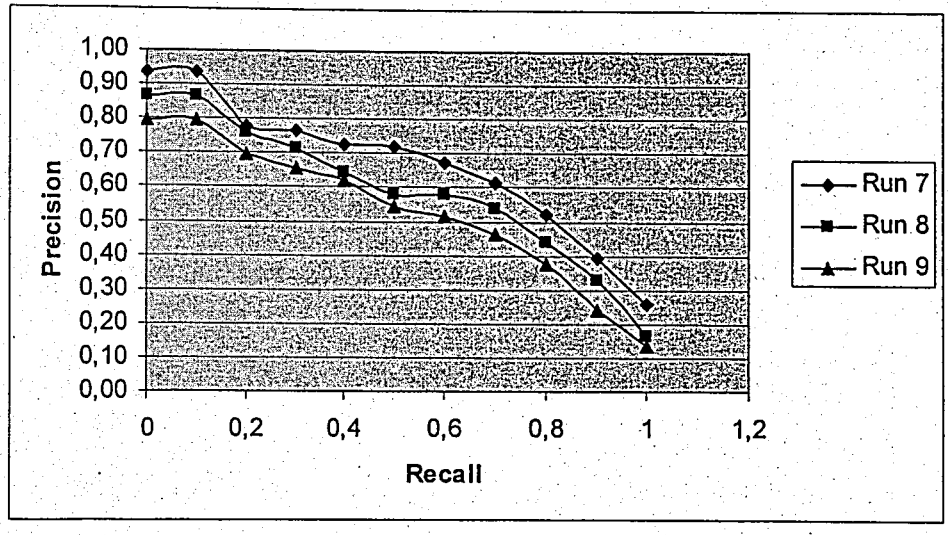


Figure 8.3. Precision-Recall graph for runs 7, 8 and 9

The relative improvement of Run 7, where syntactical processing is used, is given with respect to runs 0, 1, and 2 which are the cases where either no linguistic processing or

relatively small amount of linguistic processing (morphological and lexico-semantical processing) is used (Table 8.8).

Table 8.8. Improvement of Run 7 over runs 0, 1 and 2

Recall	Improvement of Run 7		
	Over Run 0	Over Run 1	Over Run 2
0.0	196 %	13 %	34 %
0.1	196 %	13 %	34 %
0.2	121 %	6 %	17 %
0.3	105 %	3 %	14 %
0.4	89 %	7 %	24 %
0.5	89 %	8 %	25 %
0.6	107 %	5 %	18 %
0.7	106 %	9 %	12 %
0.8	106 %	-2 %	27 %
0.9	91 %	-1 %	21 %
1.0	113 %	3 %	49 %

The recall, precision and *F* measure values are also computed at different document levels (Table 8.9). That is, the recall rate is computed for the first 10 and 20 documents in the ranked results that are returned. This gives an idea on the recall, precision and *F* measure rates in the top ranked results. Although the results are close to each other, runs 1, 4 and 7 are again better in terms of these quality measures.

Table 8.9. Average recall, precision and *F* measure values at different document levels

Run	**Recall**		**Precision**		**F-measure**	
No. of retrieved doc.s	**10**	**20**	**10**	**20**	**10**	**20**
0	0.10	0.22	0.28	0.32	0.15	0.26
1	0.24	0.38	0.74	0.58	0.36	0.46
2	0.21	0.32	0.64	0.48	0.31	0.38
3	0.20	0.33	0.62	0.53	0.30	0.41
4	0.25	0.39	0.76	0.59	0.37	0.47
5	0.20	0.32	0.62	0.48	0.30	0.38
6	0.21	0.35	0.64	0.55	0.31	0.43
7	0.25	0.40	0.78	0.61	0.38	0.48
8	0.23	0.33	0.72	0.50	0.35	0.40
9	0.23	0.37	0.70	0.58	0.34	0.45

In general, the results of the experiments suggest that the use of syntactical head-modifier pairs can improve the effectiveness of a Turkish information retrieval system by providing better content discriminators. Moreover, stemming technique is shown as highly effective for Turkish IR.

However, the use of lexico-semantical expansion, with the consideration of synonyms, hypernyms, hyponyms, holonyms and meronyms of query words resulted in degraded performance. One reason of this result may be word sense ambiguity. That is, words can be polysemous; i.e., their meaning can depend on the context [6]. Therefore, the words that are lexico-semantically related to a given word also depend on the context the word is used. For example, the word uzay (space) can be expanded as mekan (space) which is in its synset; however these two words are not always used in the same context and this can cause irrelevant retrieval results. Currently, the TURNA system does not use any word sense disambiguiation methods. Also, the lexico-semantical expansion technique should be evaluated using a much larger and representative corpus and queries.

9. CONCLUSION

In this chapter, we summarize our work and suggest directions for future improvements.

9.1. What We Did

Information retrieval presents an important application area in today's computer world. However, most of the current IR systems are still based on the old methods, mostly on the "bag of words" approach. Natural language processing presents an important potential for information retrieval due to the large amount of natural language text involved in IR.

In this thesis, we presented the design and implementation of a linguistically motivated information retrieval system for Turkish, the TURNA system, as an attempt to break out of the traditional "bag of words" approach. The system uses morphological, syntactical and lexico-semantical levels of NLP in document and query processing.

In the morphological level, we made use of the Turkish Morphological Analyzer developed by Oflazer [3]. The morphological processing is applied to documents in order to extract the stems as indexing terms. It is applied to user queries in order to extract the stems to be matched with the indexing terms of the documents in the corpus. The morphological processing is also used as input for a higher level analysis, namely the syntactical processing.

In the syntactical level, we considered a robust and efficient approach suitable to IR purposes. Keeping the phrase retrieval approach in mind, we developed a finite-state syntactical parser to identify noun phrases in the sentences of the documents and the queries. The syntactical head-modifier pairs in the noun phrases are used to improve the retrieval effectiveness.

In the lexico-semantical level, we made use of a newly developing project for Turkish; i.e., Turkish WordNet [4, 5]. We optionally applied the lexico-semantical expansion to the query terms to improve the effectiveness of the searches. In this approach, synonyms, hypernyms, hyponyms, holonyms and/or meronyms are added to the set of terms representing the query.

We represented the retrieval system in a vector space model. Three types of indexing terms are distinguished for indexing and retrieval purposes: single unprocessed words, morphological stems and syntactical head-modifier pairs.

We developed a set of Turkish documents and queries for testing the effectiveness of the TURNA retrieval system. We designed ten different experiment runs for different combinations of the processing methods. The running times were measured for morphological, syntactical and lexico-semantical level processing. The effectiveness of the system is evaluated in terms of precision and recall.

The results of the experiment suggest that the stemming technique highly improves the effectiveness of information retrieval. Although the finite-state cascade we designed is currently very limited, the syntactical head-modifier pairs have also improved the retrieval effectiveness. The improvement is even better when head-modifier pairs are heavily weighted. However, the lexico-semantical expansion technique resulted in degraded effectiveness.

9.2. Future Work

In this section, we give the points where the work reported in this thesis can be improved as future work.

One possible extension is about dealing with ambiguity. The morphological analyzer used in the morphological level outputs all the possible analyses of a given word without using any other knowledge, including contextual and statistical information. Currently, our system considers all these different morphological outputs without any ambiguity resolution. The advantage of this approach is that it does not miss correct parses. However,

the disadvantage is that, some irrelevant parses are also included. Moreover, the number of different morphological variants results in an exponential growth in the number of inputs sent to the syntactical analyzer because different morphological parses are combined to form different sentences.

One solution to this ambiguity problem may be employing a part-of-speech tagger between the morphological analyzer and the syntactical analyzer. A part-of-speech tagger can facilitate different statistical information, heuristics and constraints in order to decrease the number of alternatives sent to the syntactical parser. Some work on tagging Turkish text can be found in [28].

The syntactical level processing is an area always open to improvement. The types of noun phrases that can be identified by the finite-state parser can be increased. Currently, our syntactical parser does not recognize relative clauses because they require a more complex analysis. The syntactical parser may be improved to recognize relative clauses, and their effect on information retrieval effectiveness can be investigated.

In the search engine part, some other functionalities can be added that can be found in current commercial search engines; for example, the "exact phrase" search in Google.

The long processing time in the query matching stage, which is independent of the natural language processing techniques involved as stated in Chapter 8, can be decreased in a more realistic retrieval system.

The number and coverage of documents and queries in the corpus can be improved. Test sets can be standardized as in TREC test sets to be able to compare with related work in this area.

Techiques to solve the word sense ambiguity problem can be considered especially in the lexico-semantical expansion part as another future work.

Finally, the effect of phrases other than the noun phrase can be investigated on information retrieval; including the verb phrase.

APPENDIX A: STOP WORD LIST OF THE TURNA SYSTEM

ama	çünkü	ile	rağmen
ancak	da	ise	sadece
aynı	değil	kadar	sen
bazı	en	kim	senin
belki	fakat	kimler	siz
benim	genelde	kimse	sizin
bir	genellikle	nasıl	şimdi
biz	gibi	ne	şu
bizim	göre	neredeyse	şurada
böylece	henüz	o	ve
bu	hepsi	oldukça	veya
burada	her	onlar	yine
çeşitli	herhangi	onların	
çok	hiç	onun	
çoğunlukla	için	orada	

APPENDIX B: QUERIES USED IN THE EXPERIMENTS

Query 1

Subject : Robot technologies

Text : *Robotlarla ilgili teknolojik gelişmeler. Robot teknolojilerinin güncel uygulamaları. Robot projeleri ve pazarı. Robot araştırmaları ve geliştirme. Robot sistemlerinde ve araçlarında son yenilikler. Yapay zeka.*

Query 2

Subject : Eye diseases

Text: *Göz hastalıklarının tedavisi. Görme kayıpları. Körlüğün sebepleri. Gözün sağlığı, temizliği ve bakımı. Göz doktorları. Göz kontrol ve muayeneleri. Gözlükler ve lensler.*

Query 3

Subject : Environmental pollution

Text : *Çevrenin kirliliği. Çevre korumayla ilgili çalışmalar. Çevre örgütleri. Çevre uzmanlarının görüşleri.*

Query 4

Subject : Smoking

Text: *Sigara kullanımı. Sigaranın zararları ve sebep olduğu sağlık sorunları. Bırakma yöntemleri.*

Query 5

Subject : Depression

Text : *İnsan psikolojisi. Depresyonun belirtileri, teşhisi ve tedavisi. İlaçla ve terapiyle tedavi. Depresyonun zararları. Psikiyatrinin ve psikolojinin sunduğu çözümler.*

APPENDIX C: SYSTEM ENVIRONMENT

Our implementation is tested on a PC with the following properties:

- Operating system: Microsoft Windows XP Home Edition
- CPU: Intel Celeron 2.60 GHz
- Memory: 248 MB RAM

REFERENCES

1. Salton G. and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.

2. Google, http://www.google.com, 2004.

3. Oflazer, K., "Two-level Description of Turkish Morphology", *Literary and Linguistic Computing*, Vol. 9, No: 2, 1994.

4. Stamou, S., K. Oflazer, K. Pala, D. Christoudoulakis, D. Cristea, D. Tufis, S. Koeva, G. Totkov, D. Dutoit and M. Grigoriadou, "Balkanet: A Multilingual Semantic Network for Balkan Languages", *Proceedings of the First International WordNet Conference*, Mysore, India, 2002.

5. Bilgin, O., Ö. Çetinoğlu and K. Oflazer, "Morphosemantic Relations In and Across Wordnets: A Preliminary Study Based on Turkish", *Proceedings of the Global WordNet Conference*, Masaryk, Czech Republic, 2004.

6. Arampatzis, A., T. Weide, C. Koster and P. Bommel, "Linguistically Motivated Information Retrieval", *Encyclopedia of Library and Information Science*, Vol. 69, pp. 201-222, Marcel Dekker Inc., New York, 2000.

7. Yahoo!. http://www.yahoo.com, 2004.

8. Feldman, S., "NLP Meets the Jabberwocky: Natural Language Processing in Information Retrieval", *Online*, Volume 23, Number 3, Information Today, Inc., May 1999.

9. Arampatzis, A. T., T. Tsoris and C. H. A. Koster, "IRENA: Information Retrieval Engine Based on Natural Language Analysis", *Proceedings of RIAO'97 Computer-*

Assisted Information Searching on Internet, pp. 159-175, McGill University, Montreal, Canada, 1997.

10. Solak, A. and F. Can, "Effects of Stemming on Turkish Text Retrieval", *Proceedings of the Ninth International. Symposium on Computer and Information Sciences*, pp. 49-56, Antalya, Turkey, November 1994.

11. Çiftçi, T., *Multimedia Search Engine for Content Based Retrieval of Images and Text*, M.S. Thesis, Boğaziçi University, İstanbul, Turkey, 2002.

12. Güngör, T., *Computer Processing of Turkish: Morphological and Lexical Investigation*, Ph.D. Dissertation, Boğaziçi University, İstanbul, Turkey, 1995.

13. Evans, D. A. and C. Zhai, "Noun-Phrase Analysis in Unrestricted Text for Information Retrieval", *34th Annual Meeting of the Association for Computational Linguistics – Proceedings of the Conference*, pp. 17 – 24, 1996.

14. Zhai, C., X. Tong, N. *Milic*-Frayling and D. A. Evans, "Evaluation of Syntactic Phrase Indexing - CLARIT NLP Track Report", *The Fifth Text Retrieval Conference (TREC-5), NIST Special Publication*, pp. 347-358, 1996.

15. Strzalkowski, T. and J. P. Carballo, "Recent Developments in Natural Language Text Retrieval", *The Second Text REtrieval Conference, National Institute of Standards and Technology (NIST), Special Publication 500-215*, pp. 123-136, 1993.

16. Göçmen, E., O. Şehitoğlu and C. Bozşahin, "An Outline of Turkish Syntax", *Technical Report 95--2*, METU Department of Computer Engineering, Ankara, Turkey, 1995.

17. SIL International, *PC-KIMMO: A Two-level Processor for Morphological Analysis*, http://www.sil.org/pckimmo/about_pc-kimmo.html, 1996.

18. Phenix Spider, http://www.planet-source-code.com, 2004.

19. Jurafsky, D. S. and J. H. Martin, *Speech and Language Processing*, Prentice Hall, Inc., New Jersey, 2000.

20. Abney S., "Partial Parsing via Finite-State Cascades", *Natural Language Engineering*, Cambridge University Press, 1995.

21. Darcan, O. N., *An Intelligent Database Interface for Turkish*, M.S. Thesis, Boğaziçi University, İstanbul, Turkey, 1991.

22. Birtürk, A. and S. Fong, "A Modular Approach to Turkish Noun Compounding: The Integration of a Finite-State Model", *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium (NLPRS2001)*, Tokyo, Japan, 2001.

23. WordNet, http://www.cogsci.princeton.edu/~wn/, 2004.

24. Lee, D. L., H. Chuang and K. Seamons, "Document Ranking and the Vector-Space Model", *IEEE Software 14(2)*, pp. 67-75, 1997.

25. Ishioka, T., "Evaluation of Criteria for Information Retrieval", *IEEE / WIC International Conference on Web Intelligence, (WI 2003)*, 13-17 October 2003, IEEE Computer Society, Halifax, Canada, 2003.

26. Text REtrieval Conference (TREC) Home Page, http://trec.nist.gov/, 2004.

27. Barroso, L. A., J. Dean and U. Hölzle, "Web Search for a Planet: The Google Cluster Architecture", IEEE Micro, Vol. 23, No. 2, pp. 22-28, 2003.

28. Oflazer, K. and İ. Kuruöz, "A Tool for Tagging Turkish Text", *Technical Report, BU-CEIS-9416*, Bilkent University, Department of Computer Engineering, Ankara, Turkey, 1994.