# RETRIEVING WORDS FROM THEIR "MEANINGS"

by

ILKNUR DURGAR EL-KAHLOUT

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabanci University
Fall 2003

RETRIEVING WORDS FROM THEIR "MEANINGS"

APPROVED BY

Prof. Dr. Kemal Oflazer .....
(Thesis Supervisor)

Asist. Prof. Dr. Albert Levi .....

Asist. Prof. Dr. Uğur Sezerman .....

DATE OF APPROVAL: .....

*to my parents*

&

*to my husband, Yasser*

# Acknowledgments

I would like to express my gratitude to my supervisor Kemal Oflazer for his guidance and suggestions. I would like to thank Albert Levi, Uğur Sezerman and Berrin Yanıkoğlu for reading and commenting on this thesis. I would also like to thank Ozlem Cetinoğlu and Orhan Bilgin for their help for the Turkish dictionary and wordnet.

I am grateful to my family for their support and help thoughout my whole life. And my special thanks to my husband for his helps and encouragement while studying on his own thesis.

RETRIEVING WORDS FROM THEIR "MEANINGS"


## Abstract

The human brain is the best memory that can record and keep a huge number of information for a long time. Words, their meanings, domains, relationships between different words, and the grammars of languages are well organized in the linguistic component of brain. While speaking or writing, we can generally express our thoughts and feelings by words without thinking for a long time what the correct words can be. But, sometimes things do not go like clockwork even for human brain. In our daily life, we can often forget or not remember a word that we use frequently and exactly know its meaning. While writing a document, talking with friends, or solving a puzzle, we can not remember which word to say or to write. When we face this problem, it will be of no use to attempt searching in a traditional dictionary to find the word that we can not remember. In such cases, there is a need for resources that can locate the word from its meaning.

This thesis presents the design and the implementation of a Meaning to Word dictionary (MTW), that locates a set of Turkish words , which most closely matches the correct/appropriate one based on a definition entered by the user. The approach of extracting words from "meaning"s is based on checking the similarity between the user's definition and an entry of the Turkish dictionary without considering any semantics or grammatical information.

MTW can be used in various application areas such as computer-assisted language learning, finding the correct words for the definition questions in solving crossword puzzles, and searching the one word representations or synonyms of a multi-word definitions in a reverse dictionary.

Results on unseen data indicate that in 72% of the real users queries and 90% of different dictionaries queries, our system returns the correct answer in the first 50 results, respectively.

# Özet

ınsan beyni büyük sayıda bilgiyi kaydeden ve uzun süre bunu saklayabilen en iyi hafızadır. Kelimeler, tanımları, kullanım alanları, kelimeler arasındaki ilişkiler ve dillerin gramerleri beynin dilsel bölümünde çok iyi organize edilmiştir. Konuşurken ve yazarken, genellikle duygu ve düşüncelerimizi doğru ve uygun kelimelerin ne olduğunu üzerinde fazla düşünmeden ifade edebiliriz. Fakat, bazen işler insan beyni için bile yolunda gitmeyebilir. Günlük yaşantımızda, çok kullandığımız ve anlamını iyi bildiğimiz kelimeleri sık sık unutur, hatırlayamayız. Bir yazı yazarken arkadaşlarla konuşurken ya da bir bulmaca çözerken söyleyeceğimiz ya da yazacağımız kelimeyi bir türlü hatırlayamayız.

Böyle bir problem ile karşılaştığımızda, hatırlayamadığımız kelimeyi bulmak için klasik bir sözlüğü kullanmanın faydası olmaz. Bu gibi durumlarda, anlamından kelimeyi bulmayı sağlayacak kaynaklara ihtiyaç vardır. Bu tezde, kullanıcının tanımına dayalı en iyi uyan doğru Türkçe kelimeyi bulan "Anlamdan Kelimeye" sisteminin tasarım ve uygulanması sunulmuştur.

Anlamdan kelime çıkarma yaklaşımı, herhangi bir anlamsal ve dilbilgisel bilgi kullanmadan kullanıcının tanımı ile Türkçe sözlükteki tanımlar arasındaki benzerlikleri bulmaya dayalıdır. "Anlamdan Kelimeye" sistemi, bilgisayar destekli dil öğrenme, bulmaca çözme veya birden fazla kelimeli tanımların, bir kelime ile ifade edilebilen versiyonunu ya da eşanlamlarını öğrenmeye yarayan ters sözlükler gibi bir çok alanda uygulanabilir.

Sistem, daha önce hiç görmediği gerçek kullanıcı tanımlarında %72, farklı bir sözlükten alınan tanımlarda %90 doğru kelimeyi anılan sıraya göre ilk 50 sonuçta bulmaktadır.

# Contents

# List of Figures

# List of Tables

# RETRIEVING WORDS FROM THEIR "MEANINGS"

by

ILKNUR DURGAR EL-KAHLOUT

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabanci University

Fall 2003

RETRIEVING WORDS FROM THEIR "MEANINGS"

APPROVED BY

Prof. Dr. Kemal Oflazer .....
(Thesis Supervisor)

Asist. Prof. Dr. Albert Levi .....

Asist. Prof. Dr. Uğur Sezerman .....

DATE OF APPROVAL: .....

*to my parents*

&

*to my husband, Yasser*

# Acknowledgments

I would like to express my gratitude to my supervisor Kemal Oflazer for his guidance and suggestions. I would like to thank Albert Levi, Uğur Sezerman and Berrin Yanıkoğlu for reading and commenting on this thesis. I would also like to thank Ozlem Cetinoğlu and Orhan Bilgin for their help for the Turkish dictionary and wordnet.

I am grateful to my family for their support and help thoughout my whole life. And my special thanks to my husband for his helps and encouragement while studying on his own thesis.

RETRIEVING WORDS FROM THEIR "MEANINGS"


## Abstract

The human brain is the best memory that can record and keep a huge number of information for a long time. Words, their meanings, domains, relationships between different words, and the grammars of languages are well organized in the linguistic component of brain. While speaking or writing, we can generally express our thoughts and feelings by words without thinking for a long time what the correct words can be. But, sometimes things do not go like clockwork even for human brain. In our daily life, we can often forget or not remember a word that we use frequently and exactly know its meaning. While writing a document, talking with friends, or solving a puzzle, we can not remember which word to say or to write. When we face this problem, it will be of no use to attempt searching in a traditional dictionary to find the word that we can not remember. In such cases, there is a need for resources that can locate the word from its meaning.

This thesis presents the design and the implementation of a Meaning to Word dictionary (MTW), that locates a set of Turkish words , which most closely matches the correct/appropriate one based on a definition entered by the user. The approach of extracting words from "meaning"s is based on checking the similarity between the user's definition and an entry of the Turkish dictionary without considering any semantics or grammatical information.

MTW can be used in various application areas such as computer-assisted language learning, finding the correct words for the definition questions in solving crossword puzzles, and searching the one word representations or synonyms of a multi-word definitions in a reverse dictionary.

Results on unseen data indicate that in 72% of the real users queries and 90% of different dictionaries queries, our system returns the correct answer in the first 50 results, respectively.

# Özet

ınsan beyni büyük sayıda bilgiyi kaydeden ve uzun süre bunu saklayabilen en iyi hafızadır. Kelimeler, tanımları, kullanım alanları, kelimeler arasındaki ilişkiler ve dillerin gramerleri beynin dilsel bölümünde çok iyi organize edilmiştir. Konuşurken ve yazarken, genellikle duygu ve düşüncelerimizi doğru ve uygun kelimelerin ne olduğunu üzerinde fazla düşünmeden ifade edebiliriz. Fakat, bazen işler insan beyni için bile yolunda gitmeyebilir. Günlük yaşantımızda, çok kullandığımız ve anlamını iyi bildiğimiz kelimeleri sık sık unutur, hatırlayamayız. Bir yazı yazarken arkadaşlarla konuşurken ya da bir bulmaca çözerken söyleyeceğimiz ya da yazacağımız kelimeyi bir türlü hatırlayamayız.

Böyle bir problem ile karşılaştığımızda, hatırlayamadığımız kelimeyi bulmak için klasik bir sözlüğü kullanmanın faydası olmaz. Bu gibi durumlarda, anlamından kelimeyi bulmayı sağlayacak kaynaklara ihtiyaç vardır. Bu tezde, kullanıcının tanımına dayalı en iyi uyan doğru Türkçe kelimeyi bulan "Anlamdan Kelimeye" sisteminin tasarım ve uygulanması sunulmuştur.

Anlamdan kelime çıkarma yaklaşımı, herhangi bir anlamsal ve dilbilgisel bilgi kullanmadan kullanıcının tanımı ile Türkçe sözlükteki tanımlar arasındaki benzerlikleri bulmaya dayalıdır. "Anlamdan Kelimeye" sistemi, bilgisayar destekli dil öğrenme, bulmaca çözme veya birden fazla kelimeli tanımların, bir kelime ile ifade edilebilen versiyonunu ya da eşanlamlarını öğrenmeye yarayan ters sözlükler gibi bir çok alanda uygulanabilir.

Sistem, daha önce hiç görmediği gerçek kullanıcı tanımlarında %72, farklı bir sözlükten alınan tanımlarda %90 doğru kelimeyi anılan sıraya göre ilk 50 sonuçta bulmaktadır.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# INTRODUCTION

## 1.1 Motivation

The human brain is the best memory that can record and keep a huge number of information for a long time. Everyday, the recieved information from the environment is processed, and stored in relevant parts of the brain. The linguistic component of the brain is one of the well-developed ones. Words, their meanings, domains, relationships between different words, and the grammars of languages are learned. An avarage human being keeps a large vocabulary in his memory and, remembers any of the words and uses it whenever he needs.

While speaking or writing, we can generally express our thoughts and feelings by words without thinking for a long time what the correct words can be. The process of calling out the words from the memory to tongue takes less than a few tenths of a second. But, sometimes things do not go like clockwork even for human brain. In our daily life, we can often forget or not remember a word that we use frequently and exactly know its meaning. This can occur during any of our activities. While writing a document, talking with friends, or solving a puzzle, we can not remember which word to say or to write. Sometimes, we can not remember the name of an object that we see, even if we know its name very well.

When we face this problem, it will be of no use to attempt searching in a tradi-

tional dictionary to find the word that we can not remember. In such cases, there is a need for resources that can locate the word from its meaning. Extraction of words from "meaning"s is an application that serves users, who can not remember or want to learn a single-word version of a multi-word description.

There are much reasons we attack the problem of extracting words from "meaning"s to words for Turkish are manifold: Although some "meaning" to word reverse dictionaries are available for English in print [1,2] and computer-based [3]. No such tool is present for Turkish. Turkish posses challenging issues not encountered in other meaning to word applications, and therefore, understanding and solving the problem of extraction words from meanings for Turkish is itself an interesting research issue. This thesis presents the design and the implementation of a Meaning to Word dictionary (MTW), that locates a set of Turkish words, which most closely matches the correct/appropriate one based on a definition entered by the user. The approach of extracting words from meanings is based on checking the similarity between the user's definition and an entry of the Turkish dictionary without considering any semantics or grammatical information.

MTW can be used in various application areas such as computer-assisted language learning, finding the correct words for the definition questions in solving crossword puzzles, and searching the one word representations or synonyms of a multi-word definitions in a reverse dictionary.

MTW is trained with 50 definitions from real users and 50 definitions from a different Turkish dictionary [5] and tested with another set of 50 definitions from users and dictionary. Results on unseen data indicate that in 66% of the real users queries and 90% of different dictionaries queries, our system returns the correct answer in the first 50 results. Additionally, the conclusion offers an analysis of the system's achievements and the areas for improvements for the system.

## 1.2 Outline of the Thesis

The organization of this thesis is as follows: Chapter 2 explains the problem with detailed examples and contrasts it with information retrieval. Chapter 3 presents the main natural language processing techniques in the context of information retrieval and explains the algorithm that is developed for the problem, the applied methods and evaluations of each method. Chapter 4 presents the results and conlusions follow in Chapter 5.

# Chapter 2

# FROM "MEANING"S TO WORDS

This thesis attacks the problem of finding the appropriate *word (or words)*, whose meaning matches the given user's *definition* for Turkish language. In this thesis *definition* stands for the user's definition and *word* is the target, whose meaning matches the user's definition.

Suppose a user knows the definition of a word but can not remember the word itself. The user then enters a variety of contextual phrases that approximate his or her understanding of the word. In this work, both the user's definition and the dictionary entries are meaningful sequences of words that define a certain word. Some examples of user definitions and the corresponding meanings of some words that are collected during the implementation of MTW are listed below.[1]

- far
  - **User Definition:** *taşıtların ön kısmına mutlaka takılan ve gece yolu aydınlatmaya yarayan taşıt aksesuarı*
  - **Dictionary Definition:** *taşıtların ön bölümünde bulunan uzağı aydınlatan güçlü ışık verici*

[1]These definitions are elicited from SU graduate students

- akımölçer
  - **User Definition:** *akımı ölçmek için kullanılan alet*
  - **Dictionary Definition:** *elektrik akımının şiddetini ölçmeye yarayan araç, amperölçer*
- yöntem
  - **User Definition:** *izlenen metot*
  - **Dictionary Deifinition:** *bir amaca erişmek için izlenen tutulan yol, usul, sistem*
- istifa
  - **User Definition:** *çalıştığı işten kendi isteğiyle ayrılmak*
  - **Dictionary Definition:** *kendi isteğiyle görevden ayrılma*
- villa
  - **User Definition:** *yazlık büyük ev*
  - **Dictionary Definition:** *yazlıkta veya şehir dışında bahçeli ve güzel müstakil ev*
- korniş
  - **User Definition:** *perdeleri asmak için kullanılan bir tür mekanizma*
  - **Dictionary Definition:** *perdeleri asmaya yarayan tahta veya metalden araç*
- hücre
  - **User Definition:** *canlının en küçük yapı taşı*
  - **Dictionary Definition:** *ince bir zar içindeki protoplazma ve çekirdekten oluşmuş bir organizmanın yapı ve görev bakımlarından en küçük birliği, göze*

For example, s/he knows the meaning of the word *far*;

- *taşıtların ön kısmına takılan, gece yolu aydınlatmaya yarayan aksesuar*

However the actual definition of the word in the dictionary is:

- *taşıtların ön bölümünde bulunan uzağı aydınlatan güçlü ışık verici*

By using the user's definition, MTW should return the word *far* most probably at the top of the candidate list. Therefore, MTW deals with two challenging problems:

- locating words whose definitions are "similar" to the query in some sense.
- ranking the candidate words using a variety of ways.

Figure 2.1 shows the general structure of MTW system.

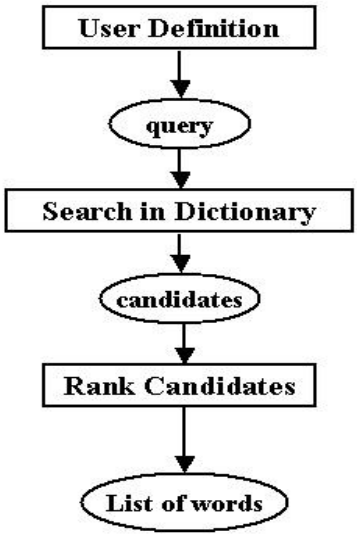


Figure 2.1: The general structure of MTW

The similarity detection is done by using symbolic and statistical methods that select terms (words, phrases, and other units) from meanings that are deemed to represent their content. User definition "resembles" a typical relevant meaning, everything about this definition becomes a valid search criterion: words, collocations,

phrases, various relationships, etc., and a query is created from the user's definition. A subsequent search process will attempt to match a preprocessed user query against term-based representations of meanings in each case determining a degree of relevance between the two. This relevance depends upon the number and types of matching terms. Simply, the meaning's relevance is a function of the number of each query word that appears in the meaning. Figure 2.2 show the process of relevance determination between the user's query and each meaning in the database.

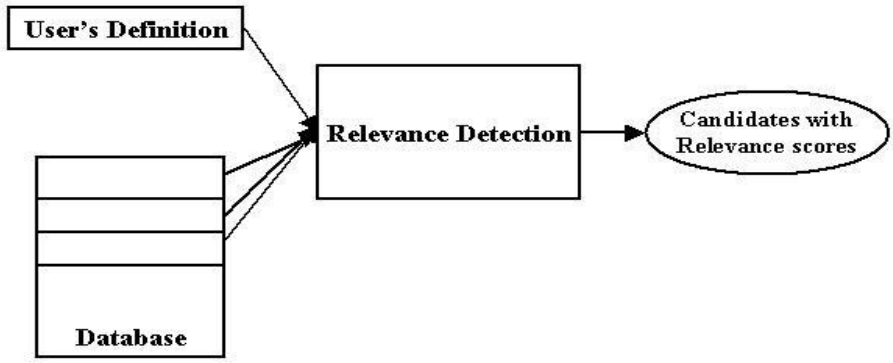


Figure 2.2: The Process of Relevance Determination

For example, for the user's definition and a word meaning from the database for the word *far*:

> **User's Definition:** *taşıtların ön kısmına takılan, gece yolu aydınlatmaya yarayan aksesuar*
> **Dictionary Definition:** *taşıtların ön bölümünde bulunan uzağı aydınlatan güçlü ışık verici*

The similar words *taşıtların, ön, bölümünde*, and *aydınlatan* are detected as *taşıtların* and *ön* directly exist in dictionary definition, the stems of *aydınlatan*

and *aydınlatmaya* are similar, and the stems of *bölümünde* and *kısmına* have same meaning. Therefore, the similarity score can be assigned 4 as the total number of similar words are 4.

For the same word *far* another :

> **User's Definition:** *taşıtların ön kısmına takılan, gece yolu aydınlatmaya yarayan aksesuar*
> **Dictionary Definition:** *bir elektrik akımının şiddetini ölçmeye yarayan araç amperölçer*

MTW is expected to compute a lower similarity score than the actual meaning's score as the meaning of the word *akımölçer* is irrelevant for the user's definiton. The similarity score of this meaning is 1 because only the word *yarayan* matches the user's definition.

Sometimes similarity matching is hard, if the user enters a definition that is too general. In such a case, the system may return many candidates, and most of them can be correct in one way or the other, or none can be correct. As an example for this case, when a user enters following definition with the intention of retrieving the word *çiçek* as:

- *bir bitki türü*

Since the user's definition is too general, the system returns thousands of candidates consisting of the word *bitki* where each of the returned words is a potential candidate for the user's definition such as:

> **abanozgiller** ⇒ iki çeneklilerden sıcak ülkelerde yetişen ve kerestesine abanoz denilen bir bitki familyası
> **abdülleziz** ⇒ akdeniz bölgesinde ve afrikada yetişen çok yıllık ve otsu bir bitki cyperus esculentus
> **acem lalesi** ⇒ taşkırangillerden turuncu ve sarı renkte çiçekli yıllık ve

çok yıllık türleri olan tohumla saksıda ve tarlada üretilebilen bir süs bitkisi güneş topu

⋮

**zülfaris zülfaruz** ⇒ baklagillerden bir süs bitkisi ve bunun güzel kokulu mor beyaz renkli saç lülesi görünüşünde olan kıvıntılı çiçeği phaseolus caracalla

The following definition for the word *gurur* is a good example for the second case:

- *insanlara ait karakter özelliği önemli bir şeye sahip oldukları ya da önemli şeyler başardıkları zaman hissettikleri duygu*

The dictionary meaning of the word *gurur*:

- *kendini beğenme büyüklenme kibir*

Although the user's definition is specific enough, the definition and the corresponding meaning have no commonality. Also, the system can not find any similar word matching the user's definiton so the words in the candidate list unluckily do not meet the user's request.

The problem of retrieving words from their meanings, at first sight, seems to be an information retrieval problem. Yet, it is distinct from information retrieval in some important aspects. The following three sections explain the nature and purpose of information retrieval, its similarities with, and its distinctions from our problem.

## 2.1 Information Retrieval (IR)

Information Retrieval is a field that uses (mostly) statistical and symbolic techniques to retrieve documents for a given query, employing little natural language analysis. Information retrieval systems use some representation of the user's query and the

documents to consider and find the subset of documents that are most relevant to the query. A typical IR system responds to the user's query by selecting documents from a database and ranking them in terms of relevance. IR must find relationships between the information needs of the users and the information held within the documents, both considered in a very general sense, and neither directly available to the computing system. People rarely supply enough information for the system to determine what the user is looking for. For example, queries for WWW search engines almost never exceed four words. An IR system well suited for general use would be able to process very short queries. The selection process in IR systems has, traditionally, relied on exact string matching. Unfortunately, such systems were not precise enough in their selections, returning too many irrelevant documents.

The basis of most IR system is a very simple but very effective approach: First, find the words in documents, and compare them to words in a query. Other types of features are often used such as phrases, named entities (people, locations, organizations), and special features (chemical names, product names) instead of words. File organizations or indexes are used for documents to increase the performance of the system. Text indexing is the process of deciding what will be used to represent a given document. These index terms are then used to build indexes for the documents. Different approaches vary in how they represent the query, the document, and how they calculate the relevance. IR usually compares query-document similarity. "Similarity" can be measured in many ways such as string matching/comparison, vocabulary used, probability that documents arise from same model, or same meaning of text. Figure 2.3 shows the flow of the retrieval process.[2]

Information retrieval [19, 23, 25] methods have been around ever since the 1950's. In early systems, a human indexed by hand, all the books in the library, assigning key words from a fixed set. (e.g., "computing", "document management"). The user could then enter queries based on these index terms (e.g., all books with key words "computing" and "document management"). The details of the relevant books would then be displayed, and possibly their location in the library. In 1960's,

[2]This figure is available on the web from CMPSCI 646 lecture notes James Allan at University of Massachusetts, Amherst

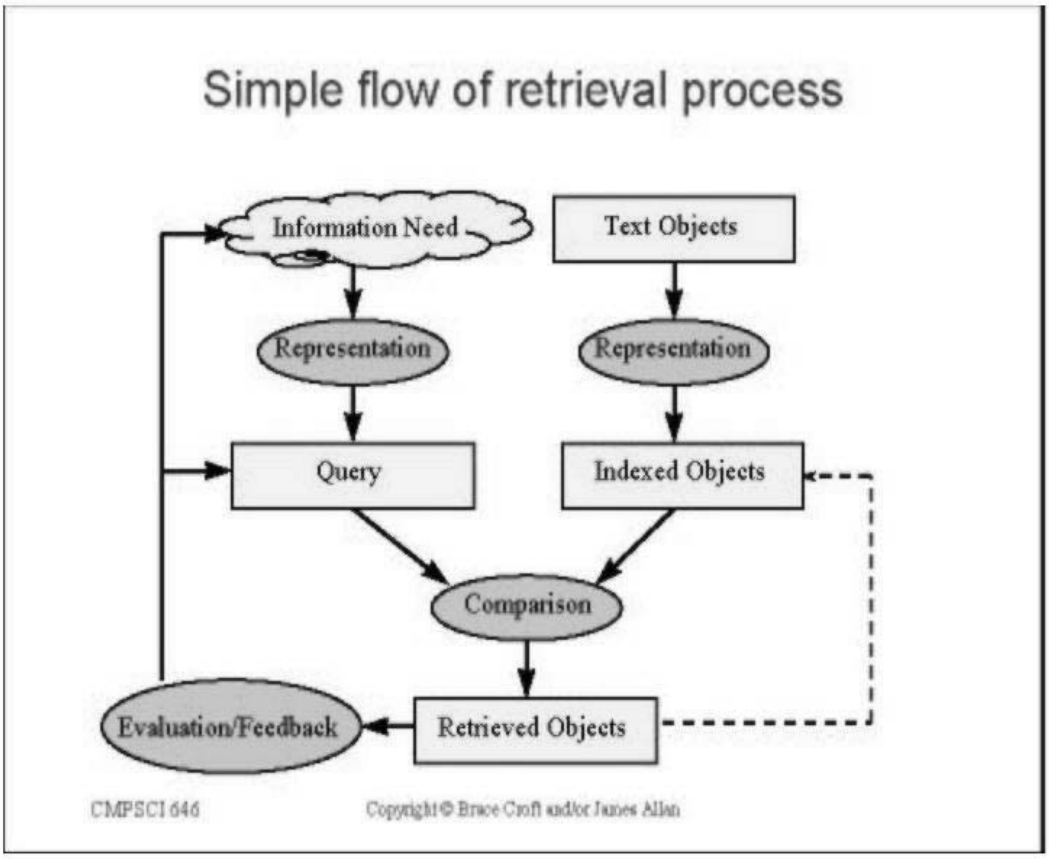


Figure 2.3: The Flow of retrieval process

basic advances in retrieval and indexing techniques started. In 1970's, probabilistic and vector space models, clustering, relevance feedback, large, on-line Boolean information services were introduced. In 1980's, simple NLP methods have been included, and IR is studied in the context. Starting in 1990's, large-scale, full-text IR and filtering experiments and systems, dominance of ranking, many Web-based retrieval engines, interfaces and browsing, multimedia and multilingual, machine learning techniques, and question answering were hot research subjects.

A retrieval model fixes the details of how we represent documents and queries, and how we compare these to find relevant documents. There are a variety of different models:

- Boolean Retrieval
- Ranked Retrieval

- Vector-space retrieval
- Probabilistic retrieval

The Boolean model of information retrieval does not use any statistics and ranking. It is important for IR because it is the first model. In Boolean Retrieval, A document is described as a set of features or index terms. e.g., "computer", "retrieval", "information", "natural language". A query is a boolean expression involving these terms. e.g., "computer AND (information OR document) AND retrieval". Documents are counted as relevant if they satisfy the boolean expression of the query. e.g., From the following set of documents, D2 and D3 would be relevant for the query "computer AND (information OR document) AND retrieval". Boolean retrieval only allows a yes/no answer concerning relevance.

D1: {computer, software, information, language}
D2: {computer, document, retrieval, library}
D3: {computer, information, filtering, retrieval}

Ranked retrieval methods allows us to rank documents in order of probability of relevance. They represent document as long term vector D giving, for every possible index term, whether it occurs in the document, e.g.:

D = {1,1,0,0,0,0,1,0,0}

This assumes fixed set of index terms, e.g.,:

{computer, software, information,...}

If the $n^{th}$ item in the term vector is 1, this means that the document contains the $n^{th}$ index term in this set. The term vector can contain weights, indicating importance of terms in the document. For example:

D = {0.4,1,0,0,0,0,0.2,0,0}

Ranked Retrieval represents query Q similarly and use some measure to determine the likely relevance of document for the given query. There are two main models for measuring relevance: Vector space retrieval [19] and probabalistic retrieval [23]. Vector Space Model [19] is one of these ranked retrieval models. It treats the index representations of documents and query to vectors in a high dimensional Euclidean space. Each term is assigned a separate dimension and the similarity measure is calculated with cosine of the angle that separates the two vectors Q (query) and D (document). *Cosine of the angle* means normalizing the vectors to the unit length and taking the vector inner product. In Vector Space retrieval, the similarity between document and Query is computed using a vector similarity function. For example, if the document and query is given as follows:

D = {1, 0.5, 0, 0.2}
Q = {1, 0, 0, 1}

Then, the calculation of similarity is;

$$sim(D, Q) = 1 \times 1 + 0.2 \times 1 = 1.2 \tag{2.1}$$

Document term weights are determined based on their frequency, *tf*, in the document, and their frequency, *df*, in the entire collection. If $N$ is the number of words in the entire collection then the weight is computed as follows: Term "computer" occurs 5 times in the document, *D1*, and 200 times in the collection. Term "bezafibrate" occurs 3 times in a the document, *D1*, and 5 times in the collection.

$$tf(t, d) = \frac{number\_of\_occurrence\_in\_document(t)}{N} \tag{2.2}$$

$$df(t) = \frac{number\_of\_occurrence\_in\_collection(t)}{N} \tag{2.3}$$

For example, If N = 1000 then tf(computer,D1) = 0.005 and df(computer) = 0.2, and tf(bezafibrete,D1) = 0.003 and df(bezafibrate) = 0.005 for terms "computer" and "bezafibrate", respectively. In WWW search engines, the weights may be calculated differently, e.g., ignoring *df* and using heuristics concerning where a term occurs in document (e.g., title).

Probabilistic Model [19, 20] are another example for ranked retrieval models. In probabalistic retrieval, relevance is viewed in probabilistic terms:

P(Relevant\Doc) (for a given query)

Then, Bayes rule and a set of independent assumptions are used, and a ranking function that computes this in terms of simpler probabilities is derived (e.g., P(Relevant\X1) where X1 is index term).

Ranked retrieval systems typically allow free text queries, such as;

- *information retrieval and filtering systems*

Common words (e.g., and) are filtered out. Then (if using vector space model), for each document, its relevance is found by adding the weights of the terms in the document vector that occur in the query. The query term 'weights' are assumed as 1 or 0. 1 indicates the existance of the word and 0 indicates the absence of the word.

The result of a ranked retrieval system is a list of documents, ranked in order of relevance. Many systems then allow an iterative process where user can mark which documents are actually most relevant, and the system tries to revise the query. This is usually done by adjusting the term weights of the query. For example, if all the documents marked as relevant include the term T, the weight of term T in the query is increased. Many methods have been used to adjust these query weights. This may be viewed in probabalistic terms, or as a machine learning problem.

There are many alternative methods for term weighting and measuring relevance. A traditional approach is for a given document collection, creates a test set consisting of queries and a human-selected set of relevant documents.

First the retrieval model is used to find the $N$ most relevant documents for each query and the "precision" and "recall" is calculated. The best retrieval model is the one giving highest precision and recall. *Recall* and *precision* measure how good a set of retrieved documents is compared with an ideal set of relevant documents.

*Recall* is the proportion of relevant documents that are retrieved and *precision* is the proportion of retrieved documents that are relevant.

### 2.1.1 Similarities between MTW and IR

This section explores the similarities of IR and MTW.

The goals of IR and MTW are similar. IR and MTW are inter-active systems that take the real user's needs and attempt to locate the most similar information that matches the user's needs. Generally, IR user enters some keywords about the topic that he wants to find. Some examples of IR queries are listed below.

- *Turizm acentaları*
- *Ístanbul iş merkezleri*
- *El işleme sanatları*

Similarly, a MTW user enters a variety of contextual phrases that approximate his or her understanding of the word. Some examples of MTW queries are listed below.

- *beklenmedik garip bir şeyin sebep olduğu şaşkınlık, şaşırma*
- *Bir kimsenin veya ailenin içinde yaşadığı yer*
- *bir işi bir ustanın yanında çalışarak öğrenen kişi*

In both of the systems, the user's request is not used directly. User requests are processed (through tokenization, stop-word removal, stemming) to determine the search terms and generate a meaningful query. Then, IR and MTW return a list of results that are expected to satisfy the user's request. Suppose, the definition entered by the user is;

- *geceleri avlanan mağaralarda yaşayan uçabilen memeli hayvan*

MTW responds to this definition with a list of words that attempts to match the user's definition;

> **samur** ⇒ kuzey avrupada yaşayan çok yumuşak ve ince tüyleri olan postu için avlanan küçük hayvan martes zibelilina
> **aslan** ⇒ kedigillerden erkekleri yeleli yırtıcı afrikada yaşayan kuyruğu ve ucu püsküllü çok koyu sarı renkli güçlü bir memeli türü arslan
> **yarasa** ⇒ yarasalardan ön ayakları perdeli kanat biçiminde gelişmiş vücudu yumuşak sık kıılarla kaplı iskeletleri hafif yapılı uçabilen memeli hayvan vespertilio
> ⋮
> **mors** ⇒ morsgillerden kuzey atlantikte yaşayan derisi dişi ve yağı için avlanan bir memeli odobenus rosmsrus
> **balina** ⇒ balinalardan yağı ve çubukları için avlanan memeli hayvan kadırga balığı falyanos balaena mistycetus

Similarly, an IR user, using the search engine Google, enters his/her needs to find the relevant documents with this topic as:

- *yarasa*

IR returns a list of document headings that are relevant to the user's topic;

- Türkiye Chiroptera-Yarasa Türleri (www.geocities.com/mammalia-2000/chirop.htm)
- Memeliler (www.biltek.tubitak.gov.tr/cocuk/01/subat/memeliler.pdf)
- Bilim ve Teknik Web Sitesi :: Canlılar dünyası (www.biltek.tubitak.gov.tr/canlilar/TR-tur-listesi/liste-memeli.htm)

⋮

IR uses collection (or collections) as the resource of search for the user's request. This collection consists of thousands of documents. MTW uses a similar resource,

the dictionary, that consists of thousands of definitions. Each document in IR collection refers to each word-meaning pair in the MTW dictionary. Like user requests, the documents in IR collection and the meanings in MTW dictionary are expressed in natural language. As the resource sizes are large, both IR and MTW do not use the documents directly. An index language is used to represent the documents or meanings to decrease the search space and increase the search time.

To find out the relevant documents, IR compares the query with each of the documents in the collection. A similar process is applied to the user definition of MTW. MTW compares the user definition and each meaning in the dictionary to match the relevant one. Generally, the systems do this comparison with string match or comparision.

Ranking the returned candidate lists in IR and MTW is the most important task. Generally, users only check the first twenty or thirty entries that are returned from the system. Therefore, returning an unordered list is inefficient as the user wants to see the most relevant information before the others. Both MTW and IR use a relevance score for each document or meaning in the collection and rank them in a decsending order. Both of the systems attempt to return the most important document or meaning at the top of the candidate list.

### 2.1.2 Differences between MTW and IR

IR seems to be similar to our system in a general frame but the question is whether IR really meets our needs or not. Our study indicates that these similarities are superficial and do not completely address our needs. Behind these surface similarities, we can actually argue that MTW is quite different from IR. Below we discuss these distinctions in detail.

Although the goals of MTW and IR seems similar as taking a query from the user and returning a ranked list of relevant information, a detailed study shows that they are different in some aspects. The IR user while entering keywords, expects to see all of the related documents about his query and most probably does not

want to rely on only one document's information. Moreover, large IR collections have many documents related to the same subject. For this reason, IR returns several documents headings that are exactly relevant in the ranked list. At this point, IR differs from MTW. The MTW user is interested in the correct word for his definition, and expects MTW to return only one relevant word when possible. Also, there can be only one (or a small number) relevant word that exactly matches the user definition in the dictionary. This is because of the structure of MTW dictionary, every meaning can be related to only one word or its synonyms. For example: if the IR user's query is *yarasa*, the system will return several documents about bats, bat's life, bat's properties, kinds of bats and so on. On the contrary, when the MTW user enters the definition;

- *geceleri avlanan mağaralarda yaşayan uçabilen memeli hayvan*

the user expects to see the word *yarasa* at the top of the ranked list.

The query expression in IR and MTW are different. The IR user enters the keywords to define his query. These keywords are generally a group of words which defines the important properties of the user's interest. As mentioned previously, the avarage number of query words in search engines such as Google does not exceed more than four or five. On the contrary, MTW needs long and detailed information as it does not perform a keyword search. In MTW, the user enters a grammatical sequence of words (a sentence or a phrase) instead of keywords to express his definition. As the user attempts to exactly define his needs, the length of the query varies and rarely reduces to less than six or seven words.

The space size of the collections and the documents is another difference of IR and MTW. As mentioned earlier, IR systems have large collections with hunderds of thousands or millions of large documents. These documents are generally long articles consisting of several paragraphs. In contrast to IR, the space size of MTW is small compared to a document database. MTW has a medium collection with thousands of small documents. MTW documents are usually one sentence long

defining a word's meaning. Table 2.1 compares some IR and MTW collection.[3] [4]

| Collection Characteristics | Cranfield | CACM | ISI | West | TREC2 | MTW |
|---|---|---|---|---|---|---|
| Collection size(docs) | 1,400 | 3,204 | 1,460 | 11,953 | 742,611 | 89,019 |
| Collection size(Mb) | 1.5 | 2.3 | 2.2 | 254 | 2,162 | 4.3 |
| Unique Stems | 8,226 | 5,493 | 5,448 | 196,707 | 1,040,415 | 21,653 |
| Stem Occurrences | 123,200 | 117,578 | 98,304 | 21,798,833 | 243,800,000 | 715,612 |

Table 2.1: Statistics of IR collections and MTW collection

Table 2.2 shows the document statistics of databases used in TREC2 and MTW collection. [5]

| Collection | Size (Mb) | # docs | Avg. # words/docs |
|---|---|---|---|
| Financial Times (FT) | 564 | 210,158 | 421.7 |
| Federal Register (FR) | 395 | 55,630 | 644.7 |
| Congressional Record (CR) | 235 | 27,922 | 1373.5 |
| FBSI | 470 | 130,471 | 543.6 |
| LA. Times | 475 | 131,896 | 526.5 |
| MTW | 4.3 | 89,019 | 7.8 |

Table 2.2: Document statictics of TREC2 collection and MTW collection

Lastly, the collections of the systems are different. Generally IR systems have collections from different sources, such as newspaper articles, papers, magazines, personal web pages, lecture notes, and so on. The collection of MTW has only one source: A set of definitions of words.

## 2.2 Applications

MTW has many applications. One application can be a Reverse Dictionary. Although there are many hard-cover [1, 2, 6, 9] and computer-based [2] applications

[3] Turkish language statistics are used for MTW

[4] The total number of definitions are presented as 'docs' in MTW

[5] The Text REtrieval Conference (TREC), co-sponsored by the National Institute of Standards and Technology (NIST) and the Defense Advanced Research Projects Agency (DARPA)

for English, a reverse dictionary is not available for Turkish. A reverse dictionary works the opposite of the traditional dictionaries [4, 5]. In traditional dictionaries, the user can find detailed definition of the word that s/he knows. For example, the user hears the word *sayfiye* and wants to learn the exact meaning. By looking at the alphabetically presented dictionary, under the letter S, the user finds the meaning:

- *yazlıkta veya şehir dışında bahçeli ve güzel müstakil büyük ev*

In the reverse dictionary, the process works in opposite fashion: The user wants to learn if s/he can express *yazlık büyük ev* in only one specific word. By using the system, user obtains the answer *villa* or *sayfiye*.

Casey's SnowDay Reverse Dictionary [3] uses n-gram analysis to determine matches between a query (the definition that the user types in) and the definitions in the dictionary. N-gram analysis is a method of matching documents based on the statistical similarity of occurences of n-grams (n-length combinations of letters) in the text. The similarity of two documents can be determined by looking at how many of their n-grams match.

Another application of MTW can be crossword puzzle solving. Puzzle solving systems [7–10] for English generally contain a huge size of vocabulary opposed to a dictionary. [10] Puzzle solving is a classical constraint satisfaction problem. These puzzle solvers want user to type in partial words with question marks filling in for unknown letters and return a list of words that satisfy the constraints of the user. During this process, the puzzle solver does not use any meaning information of words. For the pattern *en??????n?*, the puzzle solver will return a list of words matching the constraints as;

engrossing
engagement
encampment
enchanting
endearment

The words in the puzzle solver's list have completely different meaning. The user should decide to match the correct one by adding new constraints to his pattern. OneAcross Puzzle Solver [8] uses a kind of a meaning information similar to MTW. System wants a clue (optional) and a pattern that you can either enter the length of the word (if a clue exists) or you can type the word with question marks for unknown letters. Table 2.3 shows some examples of clues and patterns of Oneacross puzzle solver.

| Pattern | Clue |
|---|---|
| Trout Basket | 5 |
| - | ?a?t??s??ke |
| Cut | ???n |
| Scheme | r??? |

Table 2.3: Examples of clues and patterns of Oneacross puzzle solver

MTW presents a new view to puzzle solving. Crossword puzzle solving is viewed as matching the correct word to the puzzle clue instead of contraint satisfaction. With MTW, the user enters the clue of the puzzle and the other constraints such as length or known letters, if available. The constraints are not so necessary because MTW searchs the similarity of the clue with the meanings in the dictionary. These constraints can only increase the precision of the system. MTW can only solve vocabulary clues. Other clues such as special names, places or events are out of the scope of MTW puzzle solving.

# Chapter 3

# IMPLEMENTATION

## 3.1 General Structure

The scope of the current work is the implementation of the meaning to word system for Turkish language. Our approach to MTW is based on checking individual words in the user's request by making a number of analyses without taking into consideration the semantics or the context. Figure 3.2 shows the general structure of the MTW system. MTW uses advanced natural language processing techniques to enhance the effectiveness of term-based information retrieval. The backbone of the system is a search engine, augmented with various natural language processing for translating the user's information request into an effective query, that retrieves inverted files from pre-processed database, and then searches and ranks the words in response to the user queries. A user request is given as input to the program, and the program checks the words one by one to find the similar definitions containing these words. At the begining of the system, the database text is processed once to extract and analyze the terms as syntactic contexts. The user's natural language request is also processed and all indexing terms occurring in it are identified. In the user's request and database processing stage, removing the low quality terms(frequent and no-meaning words) from the queries is at least as important as adding synonyms and hyponyms. In some instances, low quality terms has to be removed before similar terms can be added to the query or else the effect of query expansion is drowned out

by the increased noise. After the query is constructed, the database search follows, and a ranked list of words is returned. It should be noted that all the processing steps are fully automated, no human intervation or manual encoding is required.

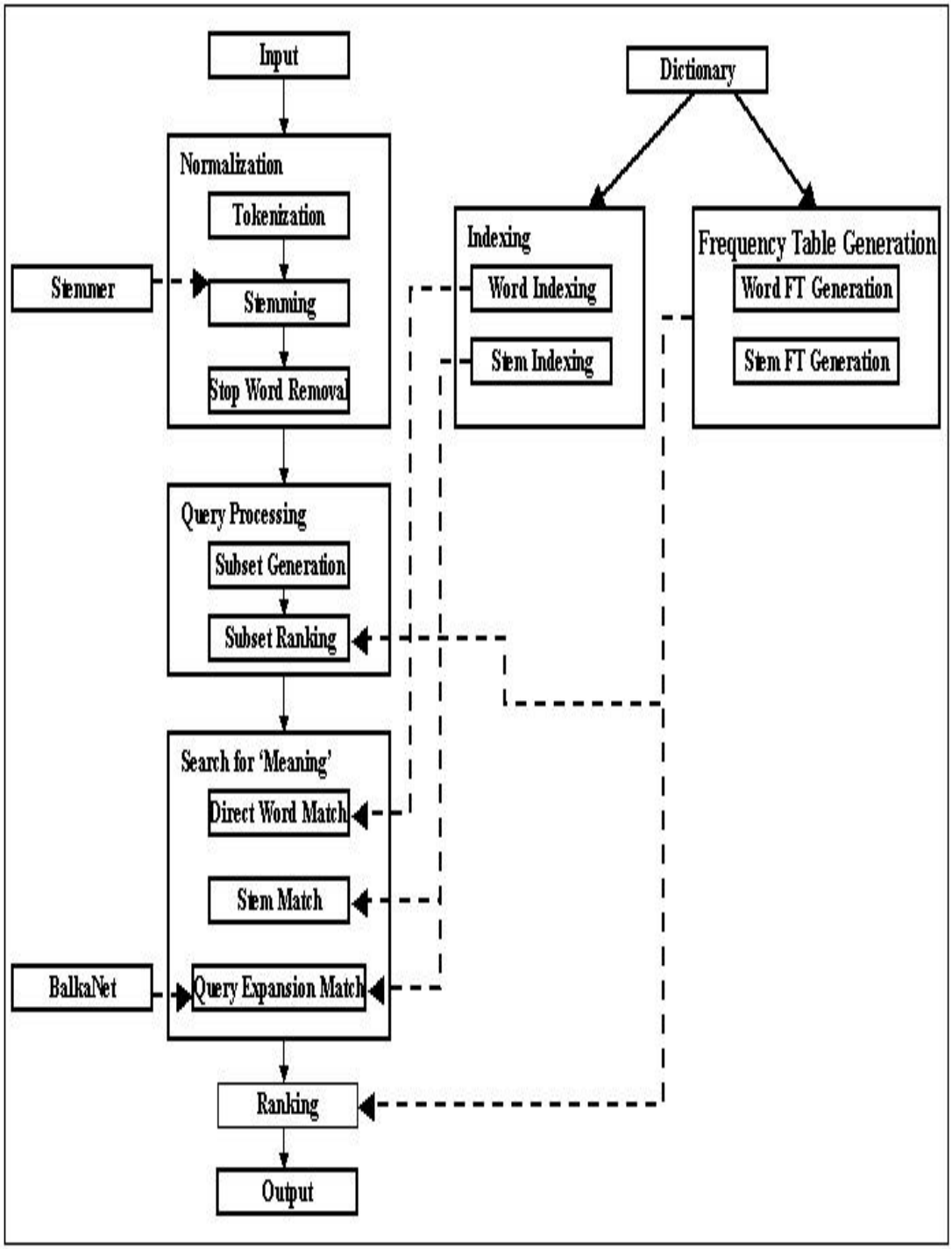


Figure 3.1: The Architecture of MTW

## 3.2 Databases and Other Sources of Information

MTW uses two sources to match the appropriate words for the user's query. These sources are the Turkish Dictionary and Turkish wordnet. Turkish Dictionary is the basic source of retrieving words from meanings. Turkish wordnet is a helper source to Turkish Dictionary to expose the relations of words in the user's request and definitions in the Turkish dictionary when needed. Following two sections explain the properties and the usages of the Turkish dictionary and Turkish wordnet in detail.

### 3.2.1 Dictionary

MTW needs a dictionary to search in and match the corresponding meanings to the user's request. MTW uses the Turkish Dictionary that is prepared and published in print and electronic copy by the Türk Dil Kurumu (TDK - Turkish Language Association). Each entry of TDK dictionary has 8 different fields indicating the word (hearword), the entry number (*entry-no*), the multiple number (*mult-no*), the sense number (*sense-no*), the type of word (*word-type*),the meaning of the word (*meaning*),the type of usage (*usage-type*) and the word's context (*context*) for each of the words. These entries shows the word, entry, multiple and sense numbers, the type of the word (noun, exclamation, adverb, adjective), its meaning, the type of usage (public, metaphor) and lastly the context (music, philosophy, theatre).

MTW uses a simplified version of the TDK dictionary as MTW pays no attention to the part-of-speech, context or sense of the word. Thus, the MTW dictionary only has alphabetically ordered words and their meanings. In the dictionary, each line contains only one word and its corresponding meaning. The Turkish dictionary used in MTW has 89,019 entries with 82,489 unique words and 21,653 unique stems. The number of unique words are less than the number of entries. TDK dictionary treats each sense of the words as a different entry. For example the word *hortum* has three different senses and each of them is treated as a different word as their meanings are different.

1- **hortum** ⇒ Filde ve bazı böceklerde boru biçiminde uzamış ağız veya burun bölümü.

2- **hortum** ⇒ Tulumba veya musluklara takılan genellikle plastikten uzun boru.

3- **hortum** ⇒ Hava veya suyun hızla dönüp sütun biçiminde yükselmesiyle oluşan, alanı dar bir siklon çeşidi.

Table 3.1 shows the general structure and some entries of the Turkish dictionary used in MTW.

Although dictionary is the main resource for searching, it is too big to search in it directly. Therefore, dictionary is processed and index and frequency tables are produced to efficient search. Further, these tables will be explained in detail in the following sections.

| Headword | entry-no | mult-no | sense-no | word-type | meaning | usage-type | context |
|---|---|---|---|---|---|---|---|
| celep | 1 | 0 | 1 | is. | koyun keçi sığır gibi kesilecek hayvanların ticaretini yapan kimse | - | - |
| evirmek | 1 | 0 | 2 | - | yapısını değiştirmek taklip etmek | - | - |
| imparator | 1 | 0 | 1 | is. | bir imparatorluğu yöneten kimse | - | - |
| ok | 1 | 0 | 1 | is. | yayla atılan ucunda sivri bir demir bulunan ince ve kısa tahta çubuk" | - | - |
| para | 1 | 0 | 1 | is. | Devletçe bastırılan üzerinde saymaca değeri yazılı kağıt veya metalden ödeme aracı nakit | - | - |
| para | 1 | 0 | 2 | - | Kuruşun kırkta biri | esk. | - |

Table 3.1: Examples from Turkish Dictionary

### 3.2.2 WordNet

WordNet [17] is an on-line lexical reference system developed in Princeton University. WordNet's design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, and adjectives are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets.

The Turkish wordnet is structured in a similar way as the WordNet around the notion of a synset, which is a set of synonymous word meanings. Two expressions are synonymous in a linguistic context C, if the substitution of one for the other in C does not alter the truth value. Table 3.2 displays some synsets from Turkish wordnet.

| Synonym Table |
|---|
| {durum, vaziyet, hal, keyfiyet} |
| {nakil, nakliyat, transfer, taşı} |
| {tavır, davranış, hareket, huy} |
| {gam, keder, hüzün, elem, üzüntü, sıkıntı} |

Table 3.2: Examples from the Synonym Table

Basic semantic relations such as hyponymy/hypernymy,antonymy and meronymy are links between these synsets. Hyponymy is a semantic relation between word meanings: e.g. *akçaağaç* is a hyponym of *ağaç*, and *ağaç* is a hyponym of *bitki*. Antonymy is a lexical relation between word forms: e.g. words *zengin* and *fakir* are antoyms. The antonym of a word x is generally not-x. Meronymy is a part-whole (or HASA) semantic relation between word meanings: e.g *araba* has *tekerlek* (as a part) or *tekerlek* is a part of *araba*. Figure **??** shows the structure of wordnet. There is also an equivalence relation for each synset to the closest concept from an Inter-Lingual-Index (ILI). The ILI contains all WordNet synsets but is extended with any other concept needed to establish precise equivalence relation across synsets.

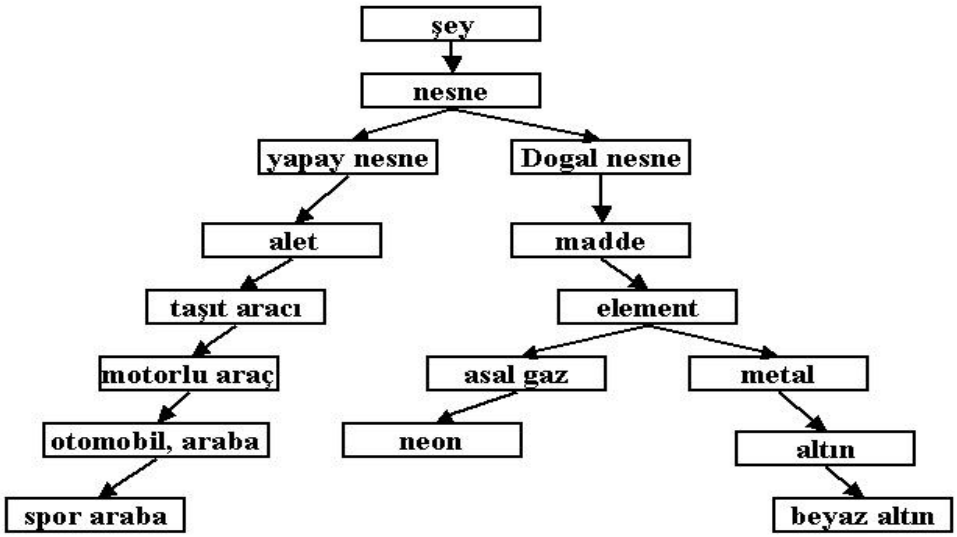


Figure 3.2: The Structure of Turkish wordnet

## 3.3 Dictionary Processing

Although the main components, dictionary and wordnet, are available in the system, MTW needs the additional information in some phases. As the dictionary is too big to traverse frequently, it is necessary to use relatively small files that are derived from the dictionary. These files contains only some important information about the dictionary. For example, while searching for relevant definitons to the user's query, it is not effective to do the search in the original dictionary. Using a small file that consists of the line numbers of each word is more effective. In addition to the line numbers, sometimes MTW needs the frequencies of the words of user definiton. Again, it is not efficient to traverse the dictionary to calculate the frequencies of words.

The following sections explain the generation of index and frequency files and additional information derived from these files.

### 3.3.1 Index Files

IR systems search thousands of documents and it is inefficient and time consuming to do the search in the original collection. Therefore, there is a need for indexing that on some means of indicates what documents are about instead of keeping the original documents. The search is done in an index file, which has a very small size with respect to the original collection.

Indexing is the base for retrieving documents that are relevant to the user's need. An index language is the language used to describe documents and requests. The elements of the index language are index terms, which may be derived from the text of the document to be described, or may be arrived at independently. Index languages for IR vary from simple to complicated ones [11–15].

Although MTW does not have a large collection, the dictionary is indexed to make the search effective within a reasonable time. Instead of complex indexing algorithms, a simple one is used because the each document of MTW is only one sentence long.

#### Word Index Table

While matching the corresponding meanings to the user request, MTW uses words as terms. Therefore, the first idea is to index the word meanings according to the words that they have. *Word Index Table* is stored in the hash table.

Indexing is done with one word by word traverse of the whole dictionary. For each line of the dictionary, the words are taken sequentially. The hash table is checked whether the word occurs or not. If the hash table does not contain this word, then a new entry is added. The word is assigned as the key of this entry and the line number is assigned as its value. If the word is already in the hash table, then its line number added to the value array. Table 3.3 shows some examples from the *Word Index Table*.

The first entry is the word, and the following numbers represent the lines (or

| Word | Lines |
|---|---|
| sepi | 70099, 70102, 74333 |
| sepici | 19384, 74324 |
| sepicilik | 19383 |
| sepicinin | 70100 |
| sepilenmek | 70103 |
| sepilenmiş | 56889, 70105 |
| sepilemek | 63906, 70101, 70104 |

Table 3.3: Examples from word index file

words) consisting of this word. For example, the stem *sepilemek* occurs in the word meanings 63906 (palamutlamak), 70101(sepileme) and 70104 (sepilenmek).

### Stem Index Table

*Stem Index Table* is generated by the help of *Word Index Table* instead of traversing the dictionary. The only difference of *Stem Index Table* from *Word Index Table* is the index terms. The index terms are the word stems instead of the original words. For generating the *Stem Index Table*, *Word Index Table* is traversed once and the indexes are stored in hash table similarly. Stemmer processes each word in the *Word Index Table* and stems are produced. Then this stem is searched in the hash table. If the stem exists in the hash table, then all the lines that this word occurs in are added to the value array of the stem. Otherwise, a new entry is added and the stem is assigned as the key and the lines that the stem occur are added as the value. Sometimes, stemmer produces more than one stem e.g. for the word *sepilenmiş* stemmer produces *sepile* and *sepi*. For such cases, the algorithm checks both of the stems and adds the lines to all of the stems as values.

Table 3.4 shows a part of the *Stem Index Table*.

The first entry is the stem, and the following numbers represent the lines (or words) coming from this stem. For example, the stem *sepile* occurs in the word meanings 70103(sepilenme), 70104 (sepilenmek), 70101 (sepileme), 70105(sepili),

| Stem | Lines |
|---|---|
| Serap | 36651, 83593, 66477 |
| Septik | 51403, 51404, 53553 |
| Sepile | 70103, 70104, 70101, 70105, 63906, 56889 |
| Sepi | 70103, 70100, 19383, 74324, 70099, 19384, 70105, 70102, 56889, 74333 |
| Senyör | 52660, 70065, 70066 |
| Sempati | 31321, 23529 |

Table 3.4: Examples from Stem Index file

63906(palamutlamak) and 56889(meşin). From Table 3.3, the exact words are *sepilenmek, sepilemek*, and *sepilenmiş*.

### 3.3.2 Frequency Tables

IR systems use term frequencies for ranking and weighting the terms in the queries and documents. In order to discriminate between good terms and poor terms, it is convenient to take into account the differences between the distribution of terms in the overall document collection and the distribution of the same terms in a set of relevant documents. IR expects that good terms will occur with a higher frequency in relevant documents than in the whole collection, and poor terms will occur with the same frequency (randomly) in both.

A similar idea is used in MTW. The relevance definition is changed a little and simplified. Because MTW has one sentence long definitions instead of long documents, it is not possible to determine the informativeness of a term in a specific meaning. But the frequency information can be used to determine the informativeness of the word in the whole collection. If a word's frequency of the user's request is high in the collection, then this word does not give much information because it occurs in many meanings. Otherwise, the word will be considered a specific word as it occurs in a few documents. Two different frequency tables are generated from Turkish Dictionary: *Word Frequency Table* and *Stem Frequency Table*.

### Word Frequency Table

The first aim of generating the *Word Frequency Table* is the determination of informative words as mentioned before. *Word Frequency Table* is generated by the help of *Word Index File*. For each of the words in *Word Index File*, the line numbers are calculated and recorded in *Word Frequency File*. Table 3.5 shows a part of the *Word Frequency Table*.

| Word | Frequency |
|---|---|
| çalıştırmadan | 1 |
| kul | 25 |
| yap | 4 |
| yapı | 194 |
| yapış | 8 |
| küçük | 1028 |
| bir | 19901 |
| bul | 1 |

Table 3.5: Examples from word-frequency table

If MTW is in Direct Word Match Mode then Subset Generation and Sorting uses *Word Frequency Table*. Subset Generation and Sorting will be explained in detait in the following sections.

### Stem Frequency Table

The generation of the *Stem Frequency Table* is similar to the *Word Frequency Table*. *Stem Frequency Table* is generated by the help of *Stem Index File*. For each word in *Stem Index Table*, the line numbers are summed and the result is recorded in *Stem Frequency Table*.

In addition to Subset Generation, stem frequencies are also important for Stop Word Removal. Table 3.6 shows a part of the stem frequency table.

The first entry of Table 3.6 is the stem and the second entry is its frequency.

| Stem | Frequency |
|---|---|
| kul | 2792 |
| bul | 3051 |
| yap | 6283 |
| gel | 2815 |
| küçük | 1070 |
| antreman | 3 |
| antrenör | 4 |
| antitoksin | 2 |
| antitez | 2 |

Table 3.6: Examples form stem-frequency table

Table 3.6, shows that the word *antitez* is more informative than the word *bul*.

If MTW is in the Stem Match Mode or Query Expansion Mode, then it uses *Stem Frequency Table* in Subset Generation and Sorting. *Stem Frequency Table* is also used in generating the Stop Word List.

### 3.3.3 Stop Word List

*Stop Word List* contains the most frequent or function words. To generate a stop word list, two different methods are used. Firstly, *Stem Frequency Table* is sorted in an ascending order and the first hundred words are taken as frequent and added to the *Stop Word List*. Secondly, the words that have no meaning such as prepositions, exclamations are determined and added to the *Stop Word List*. *Stop Word List* is used in the Stop Word Removal to eliminate the non-informative words from the user request. Table 3.7 shows some of the stopwords.

| List of Stopwords |
|---|
| iş |
| bir |
| ol |
| o, bu, şu |
| de, da, den, dan |
| veya, ya da |
| şey |
| ve |

Table 3.7: Examples from stopwords

## 3.4 Query Generation

NLP techniques are often applied to enable users to enter a natural language request without bothering them with formalisms such as Boolean connectives. In IR systems, NLP is used to preprocess the documents in order to extract content carrying terms, discover inter-term dependencies and build a conceptual hierarchy specific to the database domain, and to process user's natural language requests into effective search queries. The NLP techniques presented in this section result in a representation of dictionary and user request that is closer to the actual meaning of the text, ignoring as many of the irregularities of natural language as possible.

A typical approach to dictionary indexing and query processing is the following. First a tokenization process takes place to extract the tokens, then words are stemmed, and finally the stop words are removed.

MTW uses a similar approach for query processing. Firstly, the terms are selected and tokenized. Then, different morphological forms of words are normalized to a stem, so that words such as *kişiler* and *kişi* can be related. Finally, semantically irrelevant words that only add clutter, such as prepositions, conjuctions and pronouns and high frequency words are eliminated. While these function words are important for proper analysis of complete sentences, they can be safely removed as single-word size tokens are used in MTW.

MTW Process has three steps to generate a useful/meaningful search query: Tokenization, Stemming, and Stop Word Removal.

### 3.4.1 Tokenization

In IR systems, single terms are used for creating document index and query [24]. The words in documents and the user's natural language query words are individual terms of the vectors. Single terms are insufficient to express the whole meaning of the document or query, so different multiple term techniques have been in later IR systems.

Tokenization approach used in MTW divides the symbols into two parts: Word symbols and non-word symbols. Characters other than letters and digits are non-word symbols. Non-word symbols are eliminated from the definition because they are unnecessary in further processing. To standardize the words, all uppercase word symbols are changed to lowercase symbols.

For example, the following definition:

- *Satılan bir malın; cinsini, miktarını, fiyatını ve toplam tutarını belirten ve satıcı tarafından düzenlenerek alıcıya verilen belge*

is transformed to:

- *satılan bir malın cinsini miktarını fiyatını ve toplam tutarını belirten ve satıcı tarafından düzenlenerek alıcıya verilen belge*

after tokenization.

### 3.4.2 Stemming

Generally, the words of the user's definition and the corresponding definition in the dictionary may have the same stem but different affixes. Stemming enables matching

different morphological variants of the original definition's words and it is a kind of normalization [22, 24]. Figure 3.3 shows an example of words that have same stems.

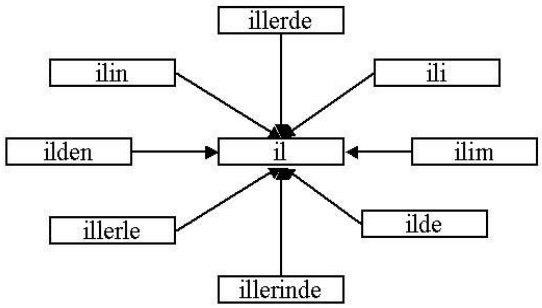


Figure 3.3: Different variants of *il*

A stemmer applies morphological rules to normalize the words. Although stemming helps to match the morphological variants, it also can hampe the retrieval. Sometimes stemming algorithms can produce different meaning stems for a word, such as for the query *en yüksek yer*, the stemmer gives two different stems *yük* and *yüksek* for the word *yüksek*; and for the query *evlenmemiş kadın*, the stemmer gives *kadı* and *kadın*, for the word *kadın*; where *yüksek* and *kadın* are the correct stems for the queries respectively. In these cases, the user questions why some offered words are candidates that were not entered. A solution to this problem can be the part-of-speech determination; a lemmatizer can find the correct lemma before the stemming process. Another approach for finding morphological variants is to generate all possibilities for the query and not to change the dictionary meanings. For example, the following tokenized definiton:

- *satılan bir malın cinsini miktarını fiyatını ve toplam tutarını belirten ve satıcı tarafından düzenlenerek alıcıya verilen belge*

is transformed to:

- *sat bir mal cins miktar fiyat ve toplam tutar belirt ve sat taraf düzenle al ver belge*

after the stemming process.

### 3.4.3 Stop Word Removal

Stop words are words that contribute nothing or very little meaning; they should be removed from the query and dictionary (or meanings) [24]. The user's definition may contain such words. A word can be a stop word from two different views. If a word occurs frequently in a dictionary, it will retrieve many meanings from the dictionary thus it is not an informative word. The selection of stop words based on frequency is simple: the top 200-300 frequent words in the dictionary are selected as stop words and removed from meanings and query. In our dictionary, words such as *iş, bir*, and *ol* are examples for most frequently occurring words.

*Stem Frequency Table* is used for stop word removal as *Word Frequency Table* is not suitable because of the aggulative structure of the Turkish Language. For example, the words *bir, biri, birinde, birileri* can be considered as stopwords and the frequencies of the words are 19901, 12, 4 and 2, respectively. With these frequencies, it is possible to eliminate only the word *bir*. The others is not detected as stopword although all of the words have the same stem *bir*. *Stem Frequency Table* having the frequencies of all words helps the selection of frequent words. For example, the following stemmed definition:

- *sat bir mal cins miktar fiyat ve toplam tutar belirt ve sat taraf düzenle al ver belge*

becomes:

- *sat mal cins miktar fiyat ve toplam tutar belirt ve sat taraf düzenle al ver belge*

after the stop word removal process.

If a word has little meaning conceptually, it is also a stop word. These words should be eliminated without looking at their frequency. If the frequency of this word is low, it is still dangerous to add it to the query set. A conceptual stop word can affect the ranking negatively. Words for prepositions, and determiners are examples of stop words. In Turkish, some stop words are *ve, o, bu, de, da*, and *ya da*. For example, the stemmed definition

- *sat mal cins miktar fiyat ve toplam tutar belirt ve sat taraf düzenle al ver belge*

becomes

- *sat mal cins miktar fiyat toplam tutar belirt sat taraf düzenle al ver belge*

after the stop word removal process.

## 3.5 Query Processing

In IR systems, while searching for the correct meaning for the user's request, rarely all of the query words can be matched with the relevant documents. Only some of the words can match the relevant documents while the others can not. For this reason, an approximate match is more suitable than the exact match of user's request with the dictionary meanings. Sparck-Jones, Walker, and Robertson [18] introduces a binary-independent probabilistic IR model. In this model, all of the different subsets of terms must be generated for user's query and each of the sets are treated as a different query. The system is expected to search and find relevant documents for all of the subsets. For the query ' social political', there are 4 different sets that must find their place with their Boolean representations: 'social and political' (1,1), 'social not political' (1,0), 'political not social' (0,1), 'not(social or political)' (0,0). The system do not search the set (0,0), because it is labeled as irrelevant at first. The rest of the sets can be relevant for the query. Hence, the systems searches for all of these sets.

Similarly,users do not enter all of the words of the correct definition everytime. User requests rarely contain all of the relevant words. Generally, correct meaning matches some words of the user's request, while it does not match the others. For example the word *villa*:

- **User Definition:** *yazlık büyük ev*

- **Actual Meaning:** *yazlıkta veya şehir dışında bahçeli ve güzel müstakil ev*

Only the words *yazlık* and *ev* matches the actual meaning.

Although the word *büyük* does not match any of the words in this meaning, this meaning is the correct one for the user's query. A similar idea like [18] is implemented in MTW. Subsets are generated as queries from the original query and MTW search all of these subsets.

Still, there is one point to take into account while searching subsets in the dictionary, what is the order of the generated subsets? An unordered search of the subsets gives irrelevant results. Therefore, MTW should determine the relevance of each subset and rank them in order to their relevance. For example, searching the (n-1)-word subsets before the n-word subsets is nonsense. An algorithm is implemented to rank the generated subset. The more informative subsets should be ranked at the top of subset list and they should be searched before the other subsets. Following two sections explains the Subset Generation and Subset Sorting in detail.

### 3.5.1 Subset Generation

To generate all subsets from the query, a subset generator processes the query and all $2^n - 1$ subsets of the $n$ words of the query are produced. Figure 3.4 show the structure of subset generation.

If the number of words in query increases then the number of subsets increases exponentially. This causes the increase of the search time. As one of the aims of MTW is to complete the search in a reasonable time, a threshold is introduced. After several experiments, the threshold value is assigned as eight. In another words,

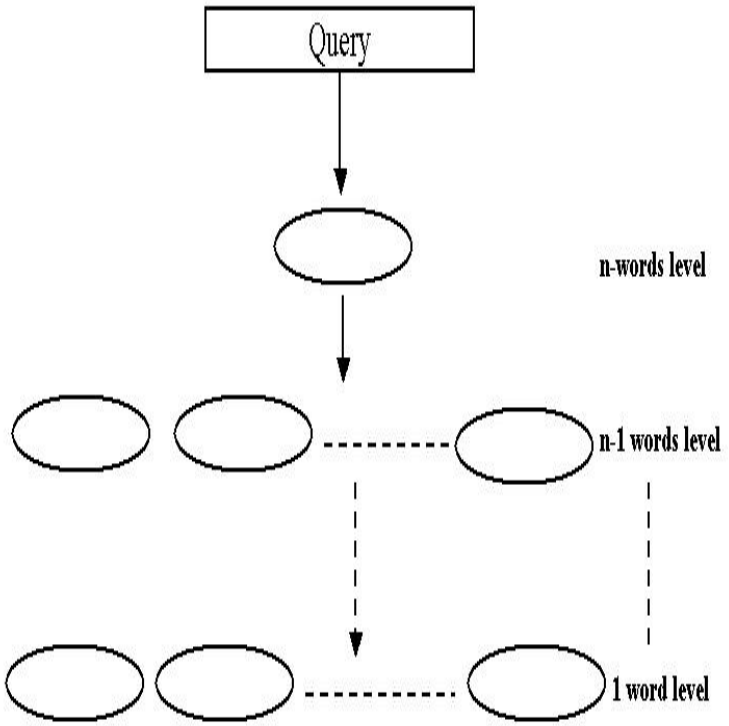


Figure 3.4: Structure of Subset Generation

MTW can use maximum eight terms from the user request. MTW sorts the terms in an acsending order by the help of *Stem Index File* and *Word Index File* and selects the most informative eight terms from the query.

A list of subsets is now available for the system:

{yazlık büyük ev, yazlık büyük, yazlık ev, yazlık, büyük ev, büyük, ev}

After generating subsets, the system searches the definitions for each subset. For example, for the subset *yazlık ev*, the system finds all words which contain the words *yazlık* and *ev* together in their meanings.

| Subset number | yazlık | büyük | ev | Generated subset |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | yazlık büyük ev |
| 2 | 1 | 1 | 0 | yazlık büyük |
| 3 | 1 | 0 | 1 | yazlık ev |
| 4 | 1 | 0 | 0 | yazlık |
| 5 | 0 | 1 | 1 | büyük ev |
| 6 | 0 | 1 | 0 | büyük |
| 7 | 0 | 0 | 1 | ev |

Table 3.8: Subset generation table for query *yazlık büyük ev*

### 3.5.2 Subset Sorting

Searching the meanings with an unordered subset list is not efficient because the system does not know which of the subsets can give the correct meaning so it searchs all of of the subsets. Assume that there 11 words in the query, then the number of the subsets is 2048. Searching all of the 2048 subset is time consuming to find the relevant word for the user's definition. With sorting, the system starts from the most informative subset that has more words than the others. A threshold is used while searching, such as 1000 candidate words, and then these candidates are sorted and the result list's first n word is printed to the user.

For the query *yazlık büyük ev*, we want to rank the word *villa*, with meaning *yazlıkta veya şehir dışında bahçeli ve güzel müstakil büyük ev* before the word *konak* with meaning *büyük ve gösterişli ev*. For this reason, the subsets are sorted in order to their number of words. The sorted list becomes:

{*yazlık büyük ev, yazlık büyük, yazlık ev, yazlık, büyük ev, büyük, ev*}

Now, it is guaranteed that the system is able to find the most similar ones before the others. But still there exists a problem. Suppose there are two words that match the same number of words in their meanings. The aim is to find the word that matches more informative words than the other definitions so the system should decide which of the subsets are more informative than others, again preserving the

first ranking criterion, the number of words of the subset [21].

There is a need for the selection of informative subsets in the same number of words group. The Word Frequency Table and Stem Frequency Table are used to determine the degree of informativeness of a word. Table 3.9 shows the frequencies of each word in the dictionary.

| Word | Word Occurrence | Stem Occurrence |
|---|---|---|
| yazlık | 9 | 12 |
| büyük | 931 | 1168 |
| ev | 157 | 734 |

Table 3.9: Frequencies of each word of the query *yazlık büyük ev*

From Table 3.9, it is easy to conclude that the word *yazlık* is more informative than the words *büyük* and *ev*, and the word *ev* is more informative than the word *büyük*. This is not surprising because the word *büyük* is a general adjective and it can be used with any noun, and the word *yazlık* is a specific kind of the word *ev*. With the help of this information, it is useful to search the word *yazlık* before *ev*, and *büyük*.

The problem for the one-word subsets is solved, this is not good to multi-word subsets. For example, it will be logical to search the subset *yazlık ev* before the subset *büyük ev*. To solve this problem, the logarithms of word frequencies are added and the result is used to define the information measure of the subset. As the frequencies of words are to small, the sum of word frequency logaritms is used instead of directly multiplying the frequencies. The sorting formula:

$$relevance\_of\_subset(j) = \frac{\sum_{i \in subset_j} log(freq_i)}{N_j} \tag{3.1}$$

where, *freq_i* is the frequency of $i^{th}$ word and $N_j$ is the number of words of the $j^th$ subset. The frequency of the word is calculated by dividing the number of occurrences by the total number of words of the dictionary as:

$$freq_i = \frac{occurrence(i)}{total\_words} \tag{3.2}$$

The *total_words* is constant for any document. For our dictionary, the total occurrence is 715,612. After sorting the subsets the last order is:

{*yazlık büyük ev, yazlık ev, yazlık büyük, büyük ev, yazlık, ev, büyük*}

After sorting the subsets, the system is ready to search for the correct word (or words) in the dictionary.

## 3.6 Searching for 'Meaning'

MTW searches all of the subsets in the dictionary. The search algorithm is as follows. Two hash tables, *Final set* and *Word Set*, are used to find the words that this subset exists in. For each subset, the search starts with the first term in the subset. The lines that the word occurs in are determined from the index tables and assigned to the Final Set. In a similar way as the *Stem Index File* and *Word Index File*, the *line number (line)* refers to each word's definition. For the following words, the lines of the word is assigned to Word Set instead of Final Set. Then Word Set and Final Set are intersected. For any line of Final Set occuring in the intersection, the line stays in the set. Otherwise, the line number is eliminated from the Final Set.

If the intersection of the *Final Set* and *Word Set* is empty, the process is terminated for this subset as there is no line containing all of these words and the process continues to search in for the following subsets. Otherwise, the process is finished and MTW takes the resultant lines to the Candidates Set.

Three different methods are implemented for MTW. In each method, either the type of terms are changed or new terms are added to the user's query.

### 3.6.1 Direct Word Match

Simplest idea for finding the similarity between two phrases is to directly match the common words of both, then decide the commonality with a threshold, and return

the best matching word.

The Direct Word Match searches the words of the query in dictionary without any process. This means that a word in the query must exists in the meaning exactly. This method is the baseline of the system. For example, the following query,

- *geceleri* ***avlanan*** *mağaralarda* ***yaşayan uçabilen memeli hayvan***

the words are taken directly as terms and the query becomes

- {*geceleri, avlanan, mağaralarda, yaşayan, uçabilen, memeli, hayvan*}

Direct Word Match searches these words in the dictionary by the help of *Word Index File* for each generated subset. The following two meanings are ranked at the top of the list:

- **Definition #1:** *yarasalardan ön ayakları perdeli kanat biçiminde gelişmiş vücudu yumuşak sık kıllarla kaplı iskeletleri hafif yapılı uçabilen* ***memeli hayvan*** *vespertilio* (word: yarasa)
- **Definition #2:** *kuzey avrupada* ***yaşayan*** *çok yumuşak ve ince tüyleri olan postu için* ***avlanan*** *küçük* ***hayvan*** *martes zibelilina* (word: samur)

The user searches for the *yarasa*, and both of the words *yarasa* and *samur* match the user request. The word *yarasa* comes from the subset {*uçabilen, memeli, hayvan*} and the word *samur* from the subset {*avlanan, yaşayan, hayvan.* Although only three words common in their meanings, these words are the most relevant words and therefore it is ranked at the top of the returned list.

### 3.6.2 Stem Match

If searching the words does not work for all cases, then searching the stems can be a solution [22]. Stem Match searches the word stems of the query, instead of the exact words, in the word meanings by the help of Stem Index file. For example:

- **Query:** ***akımı ölçmek*** *için kullanılan alet* (word: akımölçer)

- **Meaning:** *bir elektrik* ***akımının*** *şiddetini* ***ölçmeye*** *yarayan araç amperölçer*

At first sight, none of the query words is matching the correct definition and there are lots of candidates containing the word *alet*, *kullanılan*, and so on, in its definitions so the correct word is not in the returned list and all of the returned words are irrelevant for the user's query if we use Direct Word Match.

- **Query:** *akım ölç kullan alet*

- **Meaning:** *elektrik akım şiddet ölç yara araç amperölçer*

With the Stem Match, the query words are replaced with the stems, and the query is *akım, ölç, için, kullan*, and *alet*. Stem Match matches the words *akımı* and *akımının*, and the words *ölçmek* and *ölçmeye* because the words have the same common stems *akım* and *ölç*, respectively.

### 3.6.3 Query Expansion Match

Generally, for a specific word, the user's definition and the word's meaning try to express the same concept but with different words. The words of the query and the dictionary meaning are similar but expressed with different syntaxes. The users of retrieval systems (or search engines) often use different words to describe the concepts in their queries than the authors use to describe the same concept in their documents. In experiments, two people use the same term to describe an object less than 20% of the time.

For matching these kinds of words, a new method is introduced. The solution [16] is to expand the user's query with additional similar words. There are two different approaches for query expansion: Relevant Answer Expansion, and Similar Word Expansion. First one expands query with unexpanded query's relevant answers. Second one expands the query with word relations.

In Relevant Answer Expansion, firstly the original query is tried in the system, and the candidates are ranked by their relevance. Then, top scored $n$ candidates are taken for the expansion, and the relevant words of these candidates are used to expand the query. This expansion is widely used in IR systems but it is not applicable for our case. Relevant Answer Expansion uses the document frequency and term frequency to detect the relevance of a word in a document. It is not possible to find the relevant word of a sentence because the word meanings in the dictionary are only one sentence and generally each word's frequency is one.

Similar Word Expansion uses the relations between the words. The original query's words are taken and relationed words are included to the query to form the expanded one. These relations can be any relation such as synonym, hyponym, and hyperonym. Most of the retrieval systems use synonym relation alone and the results show that including other relations does not increase the precision as expected. In Query Expansion Match, the Similar Word Expansion is used to expand the query. The synonym of a word is added to the user's original query if it exists in the Synonym Table. For example:

- **Query:** *daha önce hiç evlenmemiş olan kişi*

- **Definition:** *evlenmemiş kimse*

Firstly, only the word *evlenmemiş* is matching the correct meaning. But the words *kişi* and *kimse* are similar words. By the help of Synonym Table the synonyms *insan, kimse, şahs, birey* are included the in original query and the meaning is searched with this new query. Table 3.2 shows some examples of synonyms.

## 3.7 Ranking

MTW 's second important goal is to rank the correct candidates at the top of the ranked list. MTW uses three criteria to rank the candidate definitions. Firstly, the number of matched words is calculated. Secondly, the length of the candidate defi-

nition is determined. And lastly, the longest common subsequence of the candidate definition and user definition is calculated.

### 3.7.1 Number Of Match

The number of Match criterion is simple and easy to calculate. If any meaning has more common words with the query than other meanings, then this meaning is more relevant than those in the dictionary. For example;

- **Query:** *hapishanede mahkumların kaldığı küçük odaların her biri*
- **Definition#1:** *ince bir zar içindeki protoplazma ve çekirdekten oluşmuş bir organizmanın yapı ve görev bakımlarındanen küçük birliği göze (word: hücre)*
- **Definition#2:** *hapishanede tutukluların veya hükümlülerin yalnız olarak kapatıldıkları küçük oda (word: hücre)*

In the example, the user entered a query for the word *hücre*. Suppose the above two meanings exist in the dictionary and the system wants to distinguish the similar one from other. The number of matched words is then calculated. Definition#1 matched word *küçük* so #match-1 is 1 and Definition#2 matched words*hapishane, küçük, oda* so # match-2is 3. As #match-2 > #match-1, the $2_{nd}$ meaning is more similar to the user's definition and is ranked top of the $1_{st}$ meaning.

### 3.7.2 Candidate Length

The second criterion is the length of candidate. If two candidates have the same number of matches with the user definition then, the lengths of the candidates are checked. The shorter candidate is ranked before the longer one. For example, for the following query

- *geceleri* ***avlanan*** *mağaralarda* ***yaşayan uçabilen memeli hayvan***

two candidate definitions

- **Definition #1:** *yarasalardan gelişmiş vücudu yumuşak sık kıllarla kaplı iskeletleri hafif yapılı uçabilen* ***memeli hayvan*** *vespertilio* (word: yarasa)
- **Definition #2:** *kuzey avrupada* ***yaşayan*** *çok yumuşak ve ince tüyleri olan postu için* ***avlanan*** *küçük* ***hayvan*** *martes zibelilina* (word: samur)

have the same number of matches. The length of the $2^{nd}$ definition is 16 while the length of the $1^{st}$ definition is 14. Therefore, the $1^{st}$ definition is ranked before the $2^{nd}$ definition.

### 3.7.3 Longest Common Subsequence

The third and last criterion is the length of longest common subsequence. The longest subsequence is calcuted and the definition that have longer common subsequence is ranked before the shorter ones.

We are interested in the notion of resemblance or similarity between two sentences *s1* and *s2* that have $m$ and $n$ words, respectively. A dual notion is to look at the distance between these two sentences. We are interested in a distance which enables to transform *s1* into *s2* using three kinds of basic operations: the substitution of a word of *s1* by a word of *s2*, the deletion of a word of *s1* or the insertion of a word of *s2*. A cost is associated to each of these operations and for each word of the vocabulary:

- Sub($a$,$b$) is the cost of the substitution of the word $a$ by the word $b$;
- Del($a$) is the cost of the deletion of the word $a$;
- Ins($a$) is the cost of the insertion of the word $a$.

The general problem consists of finding a sequence of such basic operations to transform *s1* into *s2* minimizing the total cost of the operations used. The total

cost is equal to the sum of the costs of each of the basic operations. This cost is a distance on the words if *Sub* is a distance on the words.

We are trying to minimize the distance between *s1* and *s2* which is generally the same (but not always) than maximizing the similarity between these two sentences. The solution is not necessarily unique. A solution can be given as a sequence of basic operations of substitutions, deletions and insertions.

For example, again for the following query

- *geceleri* ***avlanan*** *mağaralarda* ***yaşayan uçabilen memeli hayvan***

the same two candidate definitions

- **Definition #1:** *yarasalardan ayakları perdeli kanat biçiminde gelişmiş vücudu kıllarla kaplı iskeletleri hafif yapılı* ***uçabilen memeli hayvan*** *vespertilio* (word: yarasa)
- **Definition #2:** *kuzey avrupada* ***yaşayan*** *çok yumuşak ve ince tüyleri olan postu için* ***avlanan*** *küçük* ***hayvan*** *martes zibelilina* (word: samur)

This time both of the definitons match the same number of words and the length of the definitions are same. But, $1^(st)$ definition has a subsequence *uçabilen memeli hayvan* with length three while the $2^{nd}$ have two subsequences *avlanan hayvan* and *yaşayan hayvan* with lengths two. MTW ranks the $1^{st}$ definiton before the $2^{nd}$ definition.

## 3.8 Data fusion

Merging results from different systems seems to be a promising approach for achieving improved performance, since, in practice, no single information system is better than all others in all cases. Data Fusion (DF) is a broad discipline encompassing a variety of techniques for combining data, or evidence, from different sources to

achieve a unified perspective on some object, or event, of interest. DF is a problem-solving technique ased on the idea of integrating many answers to a question into a single, best answer. A number of data fusion algorithms have been proposed in distributed Information Retrieval, including min, max, average, sum, weighted average, and other linear combinations. [26–28] Among them, CombMNZ and CombSUM [26] are two well-known methods. For every document, CombSUM adds up all the scores, while CombMNZ multiplies the total scores by the non-zero scores. data fusion is used to combine the ranked lists of Stem match and Query Expansion Match. In this thesis, a simple method, weighted SUM (WSUM), [29] that allows to weight the retrieval systems is used. WSUM is a linear combination model, in which the weight of every input system is calculated via the returned results of the input systems, which is the same as the fusion process applies. Last score is calculated by:

$$Score(d) = \sum_{i} w_i * score_i(d_i) \tag{3.3}$$

where $i$ is the $i_{th}$ retrieval system. Also, a simple method is used to normalize the retrieved document scores of each retrieval system:

$$normalized\_score = \frac{unnormalized\_score - min\_score}{max\_score - min\_score} \tag{3.4}$$

# Chapter 4

# PERFORMANCE EVALUATION

MTW is implemented with Perl language as it is a strong language for string operations and pattern matching. MTW is implemented in Windows but the system also works in UNIX platform without any changes. This chapter presents the test and train results of MTW for the introduced methods Direct Word Match, Stem Match and Query Expansion in detail.

## 4.1 Setup

The experiments were carried out on two different test sets: test_set and dict_test_set. In addition, two train sets are used: train_set and dict_train_set. In the experiments 50 queries were used for each set. Queries for test_set and train_set are taken from real users. Users are given different words and asked to attempt to define these words. Queries for dict_test_set and dict_train_set are taken from a dictionary [5]. The dictionary meanings of the same 50 words that are given to the users are used as queries. Table 4.1 shows the properties of the train and test sets.

| | train_set | test_set | dict_train_set | dict_test_set |
|---|---|---|---|---|
| Number of queries | 50 | 50 | 50 | 50 |
| Avg. number of query words | 5.66 | 4.64 | 9.24 | 13.98 |
| Max. number of query words | 17 | 12 | 23 | 45 |
| Min. number of query words | 2 | 1 | 1 | 6 |

Table 4.1: Properties of test and train sets

## 4.2 Results

### 4.2.1 Direct Word Match

Table 4.2 shows the results of Direct Word Match in the test and train sets. Direct Word Match can only match 16% of the real user queries and approximately 30% of the dictionary queries. And, this method can not find any correct candidate for the 36% of the queries.

| Rank | train_set | test_set | dict_train_set | dict_test_set |
|---|---|---|---|---|
| 1 - 10 | 4 (8%) | 6 (12%) | 10 (20%) | 9 (18%) |
| 11 - 50 | 4 (8%) | 2 (4%) | 12 (24%) | 10 (20%) |
| 51 - 100 | 3 (6%) | 4 (8%) | 7 (14%) | 5 (10%) |
| 101 - 300 | 6 (12%) | 2 (4%) | 4 (8%) | 8 (16%) |
| 301 - 500 | 5 (10%) | 7 (14%) | 9 (18%) | 6 (12%) |
| 501 - 1000 | 4 (8%) | 8 (16%) | 3 (6%) | 4 (8%) |
| over 1000 | 6 (12%) | 4 (8%) | 3 (6%) | 6 (12%) |
| not found | 18 (36%) | 17 (34%) | 2 (4%) | 2 (4%) |

Table 4.2: Results of Direct Word Match

The reason of these disappointing results is discussed in the following section.

### Evaluation

Direct Word Match has advantages and disadvantages. The total number of candidates is less and the precision is high in the returned list. The method works well if the user's definition is mostly same to the word's definition. Besides these advantages, Direct Word Match lacks finding the words that have same stem. This problem occurs because of the Turkish language structure. Turkish is a aggulative language and can take infinite number of suffices after the stem. The following examples show some queries and word meanings that the method cannot match the words with same stems but with different suffices:

- **Query:** *kapı eşiklerinde bulunan ve ayakkabıların* ***altlarının*** *temizlendiği kalın bez*
- **Definition:***ayakkabıların* ***altını*** *temizlemek için kapı önlerine konulan kıl plastik vb den yapılmış yüzü tırtıklı* silecek

The common stem is *alt* for the words *altlarının* and *altını*,

- **Query:** ***akımı ölçmek*** *için kullanılan alet*
- **Definition:** *bir elektrik* ***akımının*** *şiddetini* ***ölçmeye*** *yarayan araç amperölçer*

*akım* for the words *akımı* and *akımının*, and *ölç* for the words *ölçmek* and *ölçmeye*,

- **Query:** *yeni* ***evlenen*** *kız*
- **Definition:** ***evlenmek*** *için hazırlanmış süslenmiş kız veya yeni* ***evlenmiş*** *kadın*

*evlen* for the words *evlenen, evlenmek, evlenmiş*.

### 4.2.2 Stem Match

Stem Match is tested with two different approaches. In the first one, all of the stems returned from the stemmer are included in the query. In other words, if the stemmer returns more than one stem for a word such as the stems *il* and *ilim* for the word *ilimde*, both stems are included in the query and searched in the dictionary. Table 4.3 shows the results of Stem Match with all stems included.

| Rank | train_set | test_set | dict_train_set | dict_test_set |
|---|---|---|---|---|
| 1 - 10 | 13 (26%) | 18 (36%) | 45 (90%) | 41(82%) |
| 11 - 50 | 7 (14%) | 12 (24%) | 2 (0%) | 5(10%) |
| 51 - 100 | 4 (8%) | 1 (2%) | 1 (0%) | 2(4%) |
| 101 - 300 | 3 (6%) | 3 (6%) | 2 (0%) | 1(2%) |
| 301 - 500 | 2 (4%) | 2 (4%) | 0 (0%) | 1(2%) |
| 501 - 1000 | 6 (12%) | 2 (4%) | 0 (0%) | 0(0%) |
| over 1000 | 4 (8%) | 2 (4%) | 0 (0%) | 0(0%) |
| not found | 11 (22%) | 10 (20%) | 0 (0%) | 0(0%) |

Table 4.3: Results ofStem Match with all stems included

In the second one, a simple heuristic approach is used. We assume that the longest stem returned from the stemmer is the correct stem. So, only the longest stems are included in the query this time. For example, although the stemmer returns the stems *il* and *ilim* for the word *ilimde*, only the stem *ilim* is included and searched in the dictionary. Table 4.4 shows the results of Stem Match with only longest stems included.

The results are now more encouraging than Direct Word Match. The correct words of the 60% of the real user queries and the 96% of the dictionary queries are ranked in the first 50 results.

In addition, another approach is used. This time, the frequencies of the words are included in the ranking. Table 4.5 and 4.6 show the results of Stem Match with all stems and only longest stem included, respectively.

Although the results are satisfactory enough, there is still some missing data.

| Rank | train_set | test_set | dict_train_set | dict_test_set |
|---|---|---|---|---|
| 1 - 10 | 14 (28%) | 21 (42%) | 46 (92%) | 43 (0%) |
| 11 - 50 | 5 (10%) | 9 (18%) | 1 (2%) | 5 (0%) |
| 51 - 100 | 4 (8%) | 1 (2%) | 1 (2%) | 1 (0%) |
| 101 - 300 | 3 (6%) | 1 (2%) | 2 (4%) | 1 (0%) |
| 301 - 500 | 2 (4%) | 3 (6%) | 0 (0%) | 0 (0%) |
| 501 - 1000 | 5 (10%) | 2 (4%) | 0 (0%) | 0 (0%) |
| over 1000 | 4 (8%) | 2 (4%) | 0 (0%) | 0 (0%) |
| not found | 13 (26%) | 11 (22%) | 0 (0%) | 0 (0%) |

Table 4.4: Results of Stem Match only longest stems are included

| Rank | train_set | test_set | dict_train_set | dict_test_set |
|---|---|---|---|---|
| 1 - 10 | 14 (28%) | 20 (40%) | 42 (84%) | 41 (82%) |
| 11 - 50 | 4 (8%) | 12 (24%) | 4 (8%) | 4 (8%) |
| 51 - 100 | 5 (10%) | 0 (0%) | 2 (4%) | 2 (4%) |
| 101 - 300 | 3 (6%) | 3 (6%) | 2 (4%) | 2 (4%) |
| 301 - 500 | 3 (6%) | 2 (4%) | 0 (0%) | 1 (2%) |
| 501 - 1000 | 5 (10%) | 1 (2%) | 0 (0%) | 0 (0%) |
| over 1000 | 5 (10%) | 2 (4%) | 0 (0%) | 0 (0%) |
| not found | 11 (22%) | 10 (20%) | 0 (0%) | 0 (0%) |

Table 4.5: Results of Stem Match with all stems and frequency calculation in ranking

The results and the incapabilities of the Stem Match are discussed in the following section.

### Evaluation

The Stemming helps to match morphological variants of the user's query. The words that cannot match because of suffixes are found by stemming. This increased the precision of the retrieval. As mentioned (in Stemming Section), stemming tends to hurt as many queries as it helps. Stemming sometimes mixes up two words with very different meanings to the same stem. The Table 4.7 presents some examples

| Rank | train_set | test_set | dict_train_set | dict_test_set |
|---|---|---|---|---|
| 1 - 10 | 15 (30%) | 20 (40%) | 43 (86%) | 42 (84%) |
| 11 - 50 | 3 (6%) | 11 (22%) | 3 (6%) | 5 (10%) |
| 51 - 100 | 5 (10%) | 1 (2%) | 2 (4%) | 2 (4%) |
| 101 - 300 | 4 (8%) | 3 (6%) | 1 (2%) | 1 (2%) |
| 301 - 500 | 2 (4%) | 1 (2%) | 0 (0%) | 0 (0%) |
| 501 - 1000 | 4 (8%) | 1 (2%) | 0 (0%) | 0 (0%) |
| over 1000 | 4 (8%) | 2 (4%) | 1 (2%) | 0 (0%) |
| not found | 13 (26%) | 11 (22%) | 0 (0%) | 0 (0%) |

Table 4.6: Results of Stem Match with longest stem and frequency calculation in ranking

words and stems.

| Word | Stems |
|---|---|
| yapılan | yapı yap |
| belli | bel belli |
| yazı | yaz yazı |

Table 4.7: Words and the stems returned from the stemmer

If stemmer turns more than one stem for a word, only one of these stems is the correct stem, and the rest is noise. Because it cannot be detected which of the stems is the correct one, the Stem Match uses all of the stems in the search, e.g., the system searches the stems *yaz* and *yazı* in the dictionary for the query word *yapılan*. In this case, the user cannot understand why some irrelevant word (or words) occurs in the candidate list.

Moreover, some stems have the same syntax but different meanings,e.g., the stem *yaz* means *write* as a verb and *summer* as a noun. So, the system returns the words consisting of the morphological variants of both summer and write.

Stemming increases the number of candidates because the stem frequencies is generally, the same or more than the word frequencies. This noise comes both from the multiple stems and stems that have multiple meanings.

Although Stem Match helps finding the similarity between the user's definition and meaning, it still has some weak points. The method cannot find similar words.

Query: daha önce hiç evlenmemiş olan kişi (bekar)
Meaning: evlenmemiş kimse

Query: izlenen metot (yöntem)
Meaning: bir amaca erişmek için izlenen tutulan yol usul sistem

Query: taşıtların ön kısmına mutlaka takılan ve gece yolu aydınlatmaya yarayan taşıt aksesuarı (far)
Meaning: taşıtların ön bölümünde bulunan uzağı aydınlatan güçlü ışık verici

The query - meaning couples shows the lack of the Stem Match. The words *kişi, kimse; metot, usul*; and *kısım, bölüm* are similar but the method cannot relate these words.

### 4.2.3 Query Expansion Match

We use the same approaches that are used in Stem Match. Table 4.8 and 4.9 show the results of the Query Expansion Match with all stems and only longest stems, respectively.

Although, the results do not change for the first 50 results. Query Expansion Match increased the precision of the results in the first 10 results. Now, the 48% of the real user queries and the 90 % of the dictionary queries are ranked in the first 10 results.

Similarly, the frequency calculation in ranking is applied in the Query Expansion Match. Tables 4.10 and 4.12 show the results of the Query Expansion Match

| Rank | train_set | test_set | dict_train_set | dict_test_set |
|---|---|---|---|---|
| 1 - 10 | 14 (28%) | 24 (48%) | 45 (90%) | 41 (82%) |
| 11 - 50 | 9 (18%) | 9 (18%) | 2 (4%) | 5 (10%) |
| 51 - 100 | 3 (6%) | 3 (6%) | 1 (2%) | 2 (4%) |
| 101 - 300 | 7 (14%) | 2 (4%) | 2 (4%) | 1 (2%) |
| 301 - 500 | 0 (0%) | 1 (2%) | 0 (0%) | 1 (2%) |
| 501 - 1000 | 4 (8%) | 5 (10%) | 0 (0%) | 0 (0%) |
| over 1000 | 4 (8%) | 1 (2%) | 0 (0%) | 0 (0%) |
| not found | 9 (18%) | 5 (10%) | 0 (0%) | 0 (0%) |

Table 4.8: Query Expansion Match with all stems included

| Rank | train_set | test_set | dict_train_set | dict_test_set |
|---|---|---|---|---|
| 1 - 10 | 15 (30%) | 22 (44%) | 45 (90%) | 40 (80%) |
| 11 - 50 | 8 (16%) | 12 (24%) | 1 (2%) | 5 (10%) |
| 51 - 100 | 3 (6%) | 2 (4%) | 0 (0%) | 4 (8%) |
| 101 - 300 | 6 (12%) | 2 (4%) | 3 (6%) | 1 (2%) |
| 301 - 500 | 1 (2%) | 2 (4%) | 0 (0%) | 0 (0%) |
| 501 - 1000 | 2 (4%) | 3 (6%) | 0 (0%) | 0 (0%) |
| over 1000 | 6 (12%) | 2 (4%) | 0 (0%) | 0 (0%) |
| not found | 9 (18%) | 5 (10%) | 0 (0%) | 0 (0%) |

Table 4.9: Query Expansion Match with only longest stem included

all stems, and only longest stem included with frequency calculation in ranking, respectively.

The results of frequency calculation are not as good as expected. Both the first 50 results and 10 results show a decrease in precision.

### Evaluation

Query Expansion Match has the disadvantages of Stem Mach when all of the stems are included. In addition, some noise are newly introduced because of the expansion of terms. Although the expanded terms are generally useful, sometimes the new

| Rank | train_set | test_set | dict_train_set | dict_test_set |
|---|---|---|---|---|
| 1 - 10 | 14 (28%) | 24 (48%) | 41 (82%) | 39 (78%) |
| 11 - 50 | 6 (12%) | 8 (16%) | 5 (10%) | 6 (12%) |
| 51 - 100 | 5 (10%) | 5 (10%) | 0 (0%) | 2 (4%) |
| 101 - 300 | 7 (14%) | 2 (4%) | 3 (6%) | 2 (4%) |
| 301 - 500 | 1 (2%) | 1 (2%) | 0 (0%) | 0 (0%) |
| 501 - 1000 | 5 (10%) | 3 (6%) | 0 (0%) | 0 (0%) |
| over 1000 | 3 (6%) | 2 (4%) | 1 (2%) | 1 (2%) |
| not found | 9 (18%) | 5 (10%) | 0 (0%) | 0 (0%) |

Table 4.10: Query Expansion Match with all stems and frequency calculation in ranking

terms can not match the correct definition. This causes a decrease in the ranks. Besides, only a few of the new terms are relevant to the definitions and the rest is noise. Although it has some disadvantages, Query Expansion is the best method that gives high precision in the first 10 and 50 results.

### 4.2.4 Data Fusion Results

Although Query Expansion gives better results than Direct Word Match and Stem Match, it is not efficient enough. Because of the noice introduced by the relational words, it does not work successfully for every case. In data fusion, we merged the results of Stem Match and Query Expansion. The weights of the methods are determined with training. First 1000 results from both systems are taken and tested for possible weigths. The weigths are 0.7 for Stem Match and 0.3 for Query Expansion. The results of data fusion is better for train set and test set than the results of individual methods. MTW can find the right answer of 70% of train set queries and 72% of the test set queries in the first 50 results.

| Rank | train_set | test_set | dict_train_set | dict_test_set |
|---|---|---|---|---|
| 1 - 10 | 13 (26%) | 24 (48%) | 41 (82%) | 39 (78%) |
| 11 - 50 | 8 (16%) | 7 (14%) | 4 (8%) | 6 (12%) |
| 51 - 100 | 4 (8%) | 6 (12%) | 0 (0%) | 3 (6%) |
| 101 - 300 | 6 (12%) | 4 (8%) | 4 (8%) | 2 (4%) |
| 301 - 500 | 2 (4%) | 0 (0%) | 0 (0%) | 0 (0%) |
| 501 - 1000 | 3 (6%) | 2 (4%) | 0 (0%) | 0 (0%) |
| over 1000 | 5 (10%) | 2 (4%) | 1 (2%) | 0 (0%) |
| not found | 9 (18%) | 5 (10%) | 0 (0%) | 0 (0%) |

Table 4.11: Query Expansion Match with longest stem and frequency calculation in ranking

| Rank | train_set | test_set | dict_train_set | dict_test_set |
|---|---|---|---|---|
| 1 - 10 | 13 (26%) | 24 (48%) | 41 (82%) | 39 (78%) |
| 11 - 50 | 8 (16%) | 7 (14%) | 4 (8%) | 6 (12%) |
| 51 - 100 | 4 (8%) | 6 (12%) | 0 (0%) | 3 (6%) |
| 101 - 300 | 6 (12%) | 4 (8%) | 4 (8%) | 2 (4%) |
| 301 - 500 | 2 (4%) | 0 (0%) | 0 (0%) | 0 (0%) |
| 501 - 1000 | 3 (6%) | 2 (4%) | 0 (0%) | 0 (0%) |
| over 1000 | – | – | – | – |
| not found | 9 (18%) | 5 (10%) | 0 (0%) | 0 (0%) |

Table 4.12: Data Fusion results with $w_1 = 0.7$ and $w_2 = 0.3$

## 4.3 Summary

The processing time of each query is calculated for Stem Word Match and Query Expansion Match in order to measure the speed of MTW retrieval. Timing depends on two factors. The first one is the similarity of the user definition with the dictionary definition. The number of common words decreases the processing time. The second one is the number of terms in the user definition. As mentioned before, MTW generates and searches all the subsets of terms. The increase in the number of terms increases the number of generated subsets and this effects the processing

time of the user's request in a negative way. The terms that are added to the query in Query Expansion Match increases the number of query terms and the effect of the expansion is seen in Table 4.14.

| Times | test_1 | test_2 | dict_test_1 | dict_test_2 |
|---|---|---|---|---|
| max. time(sec.) | 156 | 44 | 52 | 196 |
| min. time(sec.) | 1 | 1 | 1 | 1 |
| avg. time(sec.) | 15.4 | 5.2 | 7.5 | 18.1 |

Table 4.13: Timing in Stem Word Match

| Times | test_1 | test_2 | dict_test_1 | dict_test_2 |
|---|---|---|---|---|
| max. time(sec.) | 176 | 86 | 138 | 218 |
| min. time(sec.) | 1 | 1 | 1 | 1 |
| avg. time(sec.) | 16.8 | 9.2 | 11.3 | 23.1 |

Table 4.14: Timing in Stem Word Match

# Chapter 5

## Conclusion

In this thesis, we presented the design and implementation of a Meaning to Word system that locates a Turkish word that most closely matches the appropriate one, based on a definition entered by the user.

The performance results of MTW on unseen data from real users are rather satisfactory. Using only simple and symbolic methods, the correct words are ranked at the first 50 top results for %72 of the queries. The results on unseen queries from a different dictionary shows that the methods used while implementing MTW are reasonable.

MTW has many advantages and disadvantages. One of advantages is the free stemming and query expansion that gives a great flexibility to MTW retrieval. Some IR systems, such as search engines (Google, AltaVista) do not use the stemming and query expansion. These search engines can not match even the plural form of the word if the singular form is entered or visa versa. By stemming and query expansion in MTW, the user's definition can match the correct word(s) even if the terms of the dictionary definition does not contain the same words with same affixes. Another advantage of MTW is its speed. MTW is sufficiently speedy, although it has many time consuming components. Table 4.13 and 4.14 shows the times that MTW consumes in the test and train set queries. The maximum times are high but the avarage spent time shows that only a few querise have high timings and the rest of the queries is responded in a reasonable time.

One disadvantage of MTW is false matches. Because of the noise from the wrong stems and unrelevant synonyms, MTW can produce many irrelevant candidates. Another disadvantage is the dependence of user request. MTW works best if the request is typed similar to the actual definition. The incomplete Turkish wordnet is another disadvantage of MTW. MTW can not locate the relationships of the words of user request and dictionary definitions if the relationship does not exist in the Turkish wordnet. Lastly, the lack of opposite matching is another disadvantage of MTW. The following exaples explain this:

- *güzel olmayan*
- *tatlı karşıtı*

The words *olmayan* and *karşıtı* give a negative meaning to the queries. MTW can not detect this negativeness in the queries and searches the words *güzel* and *tatlı*. But, it should search the definitions that have the words *çirkin* and *acı, tatsız*. Therefore, the returned candidates can be totally irrelevant to the user's request unless the definition structures are similar to these definitions.

# Appendix A

### Sample Outputs of MTW

# Bibliography


[1] Kahn J.E., Readers Digest Illustrated Reverse Dictionary by Reader's Digest (1991).

[2] Edmods D.,The Oxford Reverse Dictionary, Oxford University Press (2002).

[3] Casey's SnowDay Reverse Dictionary, http://www.c3.lanl.gov/revdict (1999).

[4] Merriam-Webster's Clloegiate Dictionary, Merriam-Webster, Inc. (1998).

[5] Püsküllüğlu A., Arkadaş Türkçe Sözlük, Arkadaş Yayınevi (1995).

[6] Burger H.G., WordTree, WordTree (1984).

[7] Crossword Puzzle Solver, Hammacher Schlemmer.

[8] OneAcross, http://www.oneacross.com/crosswords (1999).

[9] Ultra Lingua Net, http://www.ultralingua.net/dictionary/#reverse (1997).

[10] Noam M. Shazeer, Michael L. Littman, Greg A. Keim, Solving Crossword Puzzles as PCS (1999).

[11] CLeverdon C.W., Mills, J. and Keen, M., Factors Determining the Performance of Indexing Systems, Vol. I, Design, Vol. II, Test Results, ASLIB Cranfield Project, Cranfield (1966).

[12] Aitchison T.M., Hall A.M., Lavelle K.H. and Tracy, J.M., Comparative Evaluation of Index Languages, Part I, Design, Part II, Results, Project INSPEC, Institute of Electrical Engineers, London (1970).

[13] Comparative Systems Laboratory, An Inquiry into Testing of Information Retrieval Systems, 3 Vols. Case-Western Reserve University (1968).

[14] Keen, E.M. and Digger, J.A., Report of an Information Science Index Languages Test, Aberystwyth College of Librarianship, Wales (1972).

[15] Salton, G., 'Automatic text analysis', Science, 168, 335-343 (1970).

[16] E.M. Voorhees, 'Using WordNet for Text Retrieval' in WordNet- An Electronic and Lexical Database, C. Fellbaum, ed. MIT Press, Cambridge, Mass.,1998, pp 285-303.

[17] G. Miller, 'WordNet: A Lexical Database for English', Communications of the ACM 38(11) pp 39-41,1995.

[18] Sparck-Jones K., Walker S., and Robertson S.E., A probabilistic model of information Retrieval: Development and comparative experiments(part 1 and 2), Information Processing & Managament 36(6),779-840 (2000).

[19] Salton G., and McHill M.J.(Eds), Introduction to Modern Information Retrieval,McGraw-Hill (1983).

[20] Maron M.E., and Kuhns J.L., On relevance, probabilistic indexing and information retrieval, Journal of the Association for Computing Machinery 7, 216-244 (1960).

[21] Salton G., and Buckley, Term weighting approaches in automatic text retrieval, Information Processing & Management 24(5), 513-523 (1988).

[22] Strzalkowski T., Natural language information retrieval, Information Processing & and Management 31(3), 397-417 (1995).

[23] Fuhr N., Probabilistic models in information retrieval, The computer Journal 35(3), 245-255 (1992).

[24] Hiemstra D., A linguistically motivated probabilistic model of information retrieval, In Proceeding of the Second European Conference on Research and Advanced Technology for Digital Libraries (ECDL), 569-584 (1998).

[25] Belkin N.J., and Croft W.B., Retrieval techniques, Annual Review of Information Science and Technology 22, 109-145 (1987).

[26] HarmanD.K., editor. Proceedings of 3rd Text Retrieval Conference (TREC-3), Gaithersburg, Maryland, USA (1995).

[27] Lee J.H., Analysis of multiple evidence combination. Procedings of the 20th Annual International ACM SIGIR Conference, 267-275, Philadelphia, Pennsylvania, USA (1997).

[28] Cottrell F. and Belew R., Automatic combination of multiple ranked retrieval systems, In Proceedings of the 17th Annual International ACM SIGIR Conference, 173-181, Dublin, Treland (1994).

[29] Wu S. and Crestani F., Data Fusion with Estimated Weights, CIKM'02, McLean, Virginia, USA (2002).

# Bibliography


[1] Kahn J.E., Readers Digest Illustrated Reverse Dictionary by Reader's Digest (1991).

[2] Edmods D.,The Oxford Reverse Dictionary, Oxford University Press (2002).

[3] Casey's SnowDay Reverse Dictionary, http://www.c3.lanl.gov/revdict (1999).

[4] Merriam-Webster's Clloegiate Dictionary, Merriam-Webster, Inc. (1998).

[5] Püsküllüğlu A., Arkadaş Türkçe Sözlük, Arkadaş Yayınevi (1995).

[6] Burger H.G., WordTree, WordTree (1984).

[7] Crossword Puzzle Solver, Hammacher Schlemmer.

[8] OneAcross, http://www.oneacross.com/crosswords (1999).

[9] Ultra Lingua Net, http://www.ultralingua.net/dictionary/#reverse (1997).

[10] Noam M. Shazeer, Michael L. Littman, Greg A. Keim, Solving Crossword Puzzles as PCS (1999).

[11] CLeverdon C.W., Mills, J. and Keen, M., Factors Determining the Performance of Indexing Systems, Vol. I, Design, Vol. II, Test Results, ASLIB Cranfield Project, Cranfield (1966).

[12] Aitchison T.M., Hall A.M., Lavelle K.H. and Tracy, J.M., Comparative Evaluation of Index Languages, Part I, Design, Part II, Results, Project INSPEC, Institute of Electrical Engineers, London (1970).

[13] Comparative Systems Laboratory, An Inquiry into Testing of Information Retrieval Systems, 3 Vols. Case-Western Reserve University (1968).

[14] Keen, E.M. and Digger, J.A., Report of an Information Science Index Languages Test, Aberystwyth College of Librarianship, Wales (1972).

[15] Salton, G., 'Automatic text analysis', Science, 168, 335-343 (1970).

[16] E.M. Voorhees, 'Using WordNet for Text Retrieval' in WordNet- An Electronic and Lexical Database, C. Fellbaum, ed. MIT Press, Cambridge, Mass.,1998, pp 285-303.

[17] G. Miller, 'WordNet: A Lexical Database for English', Communications of the ACM 38(11) pp 39-41,1995.

[18] Sparck-Jones K., Walker S., and Robertson S.E., A probabilistic model of information Retrieval: Development and comparative experiments(part 1 and 2), Information Processing & Managament 36(6),779-840 (2000).

[19] Salton G., and McHill M.J.(Eds), Introduction to Modern Information Retrieval,McGraw-Hill (1983).

[20] Maron M.E., and Kuhns J.L., On relevance, probabilistic indexing and information retrieval, Journal of the Association for Computing Machinery 7, 216-244 (1960).

[21] Salton G., and Buckley, Term weighting approaches in automatic text retrieval, Information Processing & Management 24(5), 513-523 (1988).

[22] Strzalkowski T., Natural language information retrieval, Information Processing & and Management 31(3), 397-417 (1995).

[23] Fuhr N., Probabilistic models in information retrieval, The computer Journal 35(3), 245-255 (1992).

[24] Hiemstra D., A linguistically motivated probabilistic model of information retrieval, In Proceeding of the Second European Conference on Research and Advanced Technology for Digital Libraries (ECDL), 569-584 (1998).

[25] Belkin N.J., and Croft W.B., Retrieval techniques, Annual Review of Information Science and Technology 22, 109-145 (1987).

[26] HarmanD.K., editor. Proceedings of 3rd Text Retrieval Conference (TREC-3), Gaithersburg, Maryland, USA (1995).

[27] Lee J.H., Analysis of multiple evidence combination. Procedings of the 20th Annual International ACM SIGIR Conference, 267-275, Philadelphia, Pennsylvania, USA (1997).

[28] Cottrell F. and Belew R., Automatic combination of multiple ranked retrieval systems, In Proceedings of the 17th Annual International ACM SIGIR Conference, 173-181, Dublin, Treland (1994).

[29] Wu S. and Crestani F., Data Fusion with Estimated Weights, CIKM'02, McLean, Virginia, USA (2002).