**MIDDLE EAST TECHNICAL UNIVERSITY**

**DEPARTMENT OF COMPUTER ENGINEERING**

‘Text Mining On Turkish Medical Radiology Reports’

# REQUIREMENT ANALYSIS REPORT


By


Fall, 2007

# Table of Contents

# 1. INTRODUCTION

## 1.1. Project Title

Our project title is ***RadioRead***.

## 1.2. Project Definition and Goal

In health care services, medical imaging is gaining importance nowadays. Quality of the medical images is not enough on its own for acquiring information on patients. Images need to be accurately interpreted and reported by doctors. Today, the reports of medical images are dictated as text by secretaries who listen to the tape records recorded by doctors while examining films and medical images of patients.

In current systems, most of the medical information is stored as free-text. Getting and analyzing information from a text source is more difficult than from a well structured information source. There is a need for extracting information from these text-based sources and storing the information in computationally accessible form.

It is obvious that there are plenty of documents which are kept in archives of hospitals. There are also many sources about medical situations like diseases, drugs and medical statistics on the Internet. The problem is that, nearly all of these are in textual formats. So there is a huge amount of data available which we can not benefit from with current methods in use. It is easy to access data from the Internet or from reports stored in hospital archives; but it is difficult to acquire and analyze the information enclosed inside these data.

***RadioRead*** is a project in which we will do text mining on Turkish medical radiology reports. We aim to develop a useful information acquirement method from huge amount of electronic patient reports to enable secure, ethical and user friendly access to patient information. We will provide an environment for our users to access these information as easy as using a natural language; an environment in which the user does not have to know anything about technical aspects of how the information is represented in the database systems involved. As a result; detailed information about patients can be accessed easily; more information about a patient can be given to his/her doctor before consultations; the information can be used by doctors to diagnose diseases of other patients; and statistics can be derived.

According to the market research we have done so far, we have seen that there is neither enough research nor sufficient number of production level projects for text mining in Turkish. This insufficiency is caused by the difficulty in analyzing the characteristics of the language, and also by lack of market compared to English. In this project we plan to handle usual difficulties of extracting

information from free-text clinical reports, besides providing a usable interface for different users (like doctors, assistants or statisticians) who may not have sufficient technical knowledge to use a complex program efficiently.

# 2. TEAM ORGANIZATION

## 2.1. Team Structure

We decided our team structure to be Controlled Decentralized (CD). We have a team leader who coordinates the team. Moreover, the team leader assigns tasks to group members in the team. Each member is responsible for some subtasks. The team leader is responsible for initiating new ideas, but the team takes decisions altogether and the communication among team members is very important.

## 2.2. Member Roles

The roles have been distributed among the team members as follows:

| | |
|---|---|
| **Kerem Hadımlı** | Team Leader, Initiator |
| **Çiğdem Okuyucu** | Gatekeeper, Contact Person |
| **Makbule Gülçin Özsoy** | Devil's Advocate, Recorder |
| **İpek Tatlı** | Timekeeper, Summarizer |

## 2.3. Process Model

Since we will be building an application based on research, we concluded that a spiral model, in which we will go over and over the same phases, is the most appropriate model for our project. In the project, we are required to prepare and hand-in the proposal, requirement analysis, initial design, detailed design and other documents in linear time, so we will try to keep synchronized with a waterfall model for documentation.

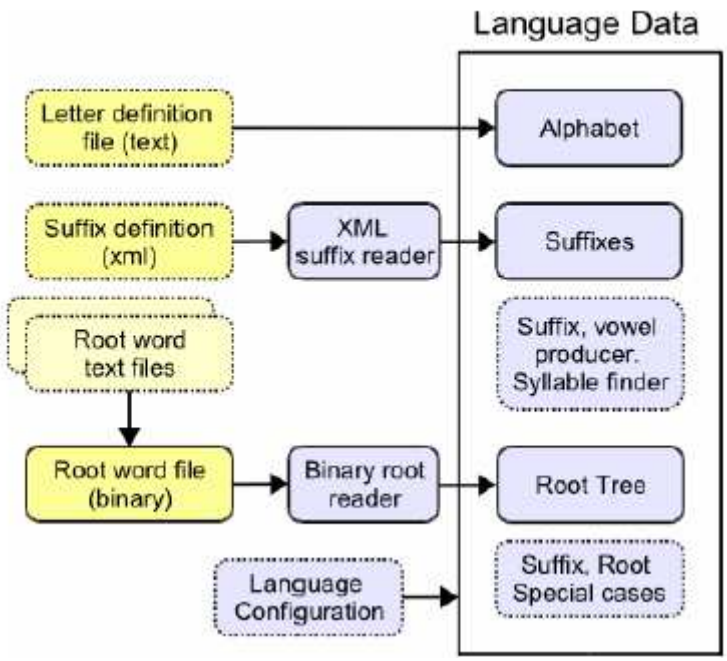
# 3. LITERATURE SURVEY

## 3.1. Zemberek

Zemberek is a generic NLP framework for morphological analysis of Turkish and some other Turkic languages. This framework is composed of *spell checking*, *morphological parsing*, *stemming*, *word construction*, *word suggestion*, *asciifier* and *deasciifier*.

Zemberek is a library which consists of 'language structure information' and 'NLP operations' parts. Core library contains NLP specific algorithms and has some tools for language implementations. In Zemberek there are some text based configuration files keeping information such as special cases, suffix mechanism etc.

Here is a schema of language information elements:




Alphabet information, letters, vowels, types of vowels, unvoiced consonants and non-ASCII characters are kept in a simple text based file. This file helps preparing and validating strings before using them in NLP.

Suffixes are defined in a special XML configuration file. This file is composed of 'suffix groups' and 'individual suffix information'. In this file each suffix is defined by an attribute. Some suffixes have suffix production elements of that suffix. Below, there is an example of a suffix file in Zemberek:

```
<ek ad="ISIM_COGUL_LER" uretim="lAr">
        <ardisil-ekler>
                <kume>ISIM_HAL</kume>
                <kume>IMEK_ZAMAN</kume>
                <aek>ISIM_SAHIPLIK_BEN_IM</aek>
                <aek>ISIM_SAHIPLIK_SEN_IN</aek>
                <aek>ISIM_SAHIPLIK_O_I</aek>
                ...
        </ardisil-ekler>
</ek>
```

Suffix objects has also a list of possible subsequent suffixes and special cases that can be attached to that suffix. Below, there are examples of a special suffix case:

```
ara → ar-ıyor          not 'ara-yor' or 'ara-ıyor'
kes → kes-il-mek
gel → gel-in-mek       not gel-il-mek
```

The meaningful part of a word without the suffixes is called “root word”. In Zemberek, root words are kept in a text based file. Root of a word, type of that word and special cases are kept in this file. This file is not used directly, but with some binary tree representations.

As mentioned above, there are some special cases for some root words. Some cases are actually parts of language but for some algorithmic reasons they are also assumed as special cases: Below there are examples:

```
saat → saatler not saatlar. Breaks vowel harmony, second 'a' is pronounced frontal
red → reddi    not redi. Last consonant repeats
burun → burnu  not burun-u. Last vowel of the root drops.
su → suyu      not sunu. This is only for word “su” - water
ben → bana     not bene. Vowel changes without a rule. This is only for first person
               subjects "ben" and "sen"
```

As seen above, special cases have some properties like if the root alters after a suffix is attached or not; if it is optional or not; if it is occurred according to the type of last letter of root or first element of suffix. When this occurs, *word modifier* is used.

Zemberek requires a *root finding mechanism*, because it uses a root dictionary based parser. A word, once read, is related to some special cases attached to its root word. This resulting object is stored in a special *direct acyclic word graph* for fast access later. When a root is added to the tree, system also finds altered states of that root in the case of whether a special case is occurred.

Standard *Morphological Parser* in Zemberek finds the possible roots and suffixes of a given word. For this job, simple dictionary based top-down parser is used. Firstly it converts the word to its lowercase. After possible root candidates are found it finds suffixes according to that root candidates and their types. Zemberek does not parse prefixes, currently.

*Spell checker* in Zemberek uses morphological parser for its job.

*Error tolerated parser* and *word suggestion* mechanism is also available in Zemberek. While it is suggesting words, results are sorted in the frequency of usages.

*Word construction mechanism* is simpler than morphological parsing. Below, there is an example:

```
for the Turkish word "koyunlara", parser produces:
Root: [koyun, NOUN]
Suffixes:[NOUN_PLURAL + NOUN_DATIVE]
We use the result in the constructor, it produces: [koyun] + [lar, a]
```

Zemberek is coded in Java for platform independency and speed. Zemberek is used not only in scientific researches, but also in real world applications, in natural language processing works, in OpenOffice, in Turkish GNU/Linux distribution Pardus and some Turkish text mining works and projects.

## 3.2. SNOMED CT – Systematized Nomenclature of Medicine, Clinical Terms

A clinical terminology is a structured collection of descriptive terms for use in clinical practice. These terms describe the care and treatment of patients and cover areas like diagnoses, symptoms, surgical procedures, treatments and drugs as well as terms used for healthcare administration. The recording of clinical data can be communicated in a standard way between healthcare systems and individuals.

SNOMED CT is a comprehensive clinical terminology that is used to code, retrieve, and analyze clinical data covering diseases, clinical findings and procedures. It has a consistent way of indexing, storing, retrieving and aggregating clinical data across specialties and sites of care. It helps to structurize and computerize the medical record, reducing the variability in the way data is captured, encoded and used for clinical care of patients and research. This terminology is developed by the NHS and the College of American Pathologists (CAP). It is the healthcare industry standard of clinical terminology. The aim of this terminology is to represent the words and phrases used in healthcare in a consistent way by using unique codes that are recognizable by machines. It consists of tree main parts: concepts, descriptions and relationships.

A concept is a clinical meaning identified by a unique numeric identifier (ConceptID) that never changes. ConceptIDs do not contain hierarchical or implicit meaning – they do not reveal any information about the nature of the concept. Each concept has one 'fully specified name' that provides a unique and unambiguous description for a concept.

In addition to the fully specified name, every concept has a number of descriptions. These can represent the terms that are in everyday use. There are often some synonyms for a single concept.

Every concept is placed in a hierarchy by which it is related to other concepts. A relationship is assigned only when that relationship is always known to be true. The relationships are used to define a concept where it can be expressed in terms of other relationships. They may also define how a concept is reasonably further refined or qualified. Relationships are very powerful mechanisms which allow not only grouping of closely related concepts, but also enabling collection of medical information for secondary purposes without any loss of the detail required for primary clinical use.

An example of the structure of a SNOMED CT concept:

```
Concept:
*ConceptID: 22298006
*Fully Specified Name: myocardial infarction (disorder)
```

```
Descriptions:
*Preferred term: myocardial infarction
*Synonym: cardiac infarction
*Synonym: heart attack
*Synonym: infarction of heart
```

```
Relationships:
*Defining relationships (IS a)
        *Concept: structural disorder of heart
                -Associated morphology: Infarct
                -Finding site: myocardium structure
        *Concept: injury of anatomical site
                -Associated morphology: Infarct
                -Finding site: myocardium structure
        *Concept: myocardial disease
                -Associated morphology: Infarct
                -Finding site: myocardium structure
*Allowable qualifiers:
        *Qualifier: onset
        *Qualifier: severity
        *Qualifier: episodicity
        *Qualifier: course
```

## 3.3. Link Grammar Parser

Link Grammar is a formal grammatical system. It is developed by Davy Temperley and Daniel Sleator. In this system, the relation between words in a sentence is gathered, and encoding natural language grammars becomes easier. Link grammar is similar to dependency-grammar which

includes a tree-like hierarchy, but in link grammar directionality in relations between words is not lost as it happens in dependency-grammar.

A "Link Grammar Parser" satisfies the following conditions when it finds links between words. [3]

***1. Planarity:*** The links do not cross.

***2. Connectivity:*** The links suffice to connect all the words of the sequence together.

***3. Satisfaction:*** The links satisfy the linking requirements of each word in the sequence.

There are some terms that are mostly used in Link Grammar parsers:

***word:*** The terminal symbols of the grammar.
***linking requirement:*** Each word has a linking requirement.
***sentence:*** A sequence of words.
***link:*** A way to draw arcs among the words so as to satisfy the conditions above.
***dictionary:*** The linking requirements of each word are contained in a dictionary.
***connector:*** Each of the intricately shaped labeled boxes (exp: D, O, S)
***linkage:*** A set of links that prove that a sequence of words is in the language of a link grammar.

"Link Grammar Parser for English" is a syntax parser which parses sentences with link grammar. This parsing program was written by Davy Temperley and Daniel Sleator using C programming language. It is licensed under the GNU General Public License. The program tries to parse the sentence using every possibility of connection using the given link grammar which is achieved by an exhaustive search algorithm. This algorithm is developed by the developers of "Link Grammar Parser for English" and it is $O(n^3)$.

The parser, "Link Grammar Parser for English", has a dictionary of about 60,000 word forms [4]. In the parser, these word forms are made up from words and their connectors. There are connectors towards left or right. These connectors labeled with "-"or "+" accordingly. A left connection connects with a right connection and forms a "link". Words have rules that show which word can be connected to another word according to the rules. If all words in a sentence are connected according to rules, we call that sentence a valid sentence.

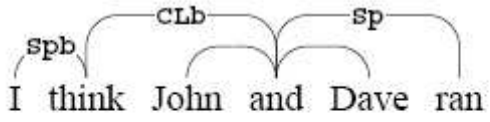
"Link Grammar Parser for English" covers large variety of words including some idioms, so the parser can handle most of the sentences in English. If it encounters unknown words, it skips them and handles the rest of the sentence. It can also handle unknown vocabulary by guessing and by

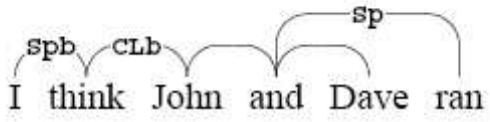
using its category of unknown words. This makes the parser dynamic, and capable of parsing sentences without errors. The parser also has knowledge of capitals, numbers and punctuations.

There is also another parser which is inspired by "Link Grammar Parser for English": "Link Grammar Parser for Turkish". "Link Grammar Parser for Turkish" is written as a part of a thesis, so it is not as advanced as "Link Grammar Parser for English". In the thesis, first the formalism of link grammar is described and some features of Turkish which do not exist in English are explained. Then new architecture of the system and some special preprocessing that is done before the parsing step is described and the link grammar specification for Turkish is presented [5]. At the end, 80% success for parsing Turkish sentences is achieved.

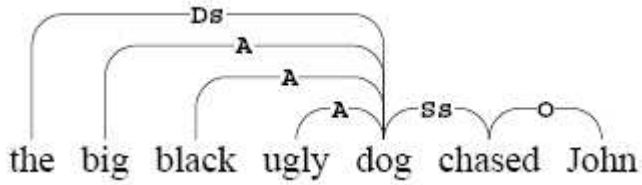
Here are some examples of parsed sentences using "Link Grammar Parser for English" and "Link Grammar Parser for Turkish":

*English-1.* "I think John and Dave run."



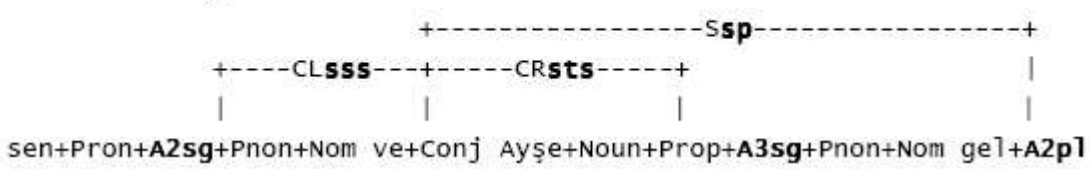




*English-2.* "The big black ugly dog chased John."



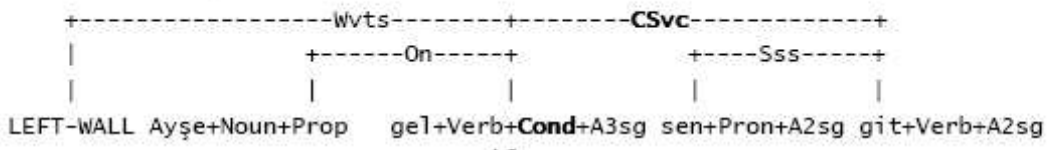

*Turkish-1.* "Sen ve Ayşe geldiniz."

Related Linkage:




*Turkish-2.* "Ayşe gelirse sen gidersin."

Related Linkage:




## 3.4. CLEF – Clinical e-Science Framework

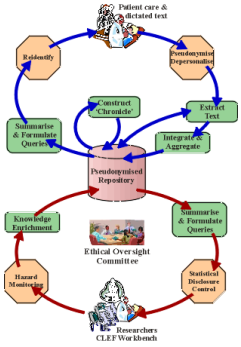
CLEF is a research project for creating a full framework that focuses on capturing, processing and analyzing information on cancer patients, and handling their care, with the use of electronic patient reports, throughout the United Kingdom. The system was funded for 3 years, and later, it would be integrated into NHS – National Health Services throughout the UK.

The CLEF project was started with the observation: Although there is a great number of data about patients, about diseases, about drugs; these data is not available for processing directly, as they're in textual formats.

CLEF aims to be a complete solution that handles capturing, processing and analyzing information and is largely scalable through the use of GRID technologies.

CLEF is founded on three circular paths:

1. Patient care and handling of individual patients' data
2. Constructing “chronicle”s out of individual patients' data, expanding the context of these data with new information from other sources
3. Allowing researchers to analyze all the data, and spot new phenomena throughout these information.




Besides CLEF's vision of having two outer circular paths on accumulated information (one for patient-doctor, and one for data-researcher relations), CLEF has a strong capability of enabling user to access clinical data naturally.

Accessing clinical data is divided into two parts:

1. Viewing data in repository
2. Querying repository for research

For viewing data, a proposed engine in CLEF system automatically transforms the computer-accessible data to a roughly generated natural-language text that is more narrative. This makes it easy for the doctor to understand the data stored.


For querying data, a proposed query-constructor exists in CLEF prototypes. The aim of this constructor is to make the researcher access the information "as easy as using a natural language". The method used is not to allow the researcher to enter free text, and then analyze the text using NLP techniques, as that might cause ambiguities. Instead, CLEF proposes a query-constructor – a natural language generator that is similar to a GUI with dropdowns in technical aspects, but seems as a natural language to the user.

## 3.5. Java Language

We did some research on the Internet to decide the programming language which we will use to develop our project RadioRead. We examined popular object oriented languages: C++, Java and .NET. All of them are high level languages and there are some similarities among them. C++ is the fastest among them but is the hardest one to implement our project in. Java is a platform-independent language. We are not familiar with .NET.

We decided on using Java for our project; because Java offers a wide range of libraries and APIs, also the native language of Zemberek library which we plan to use.

In brief, Java is good at object-oriented programming, and still not bad in performance; is platform-independent; has good integrated development environments such as Eclipse. We believe Java is the best for dealing with such a complex project.

# 4. PROJECT REQUIREMENTS

## 4.1. System Requirements

**General Aspects**

- Java as a programming language
- PostgreSQL Database Management System
- Hibernate library may be considered for persisting Java objects directly in DBMS
- Zemberek library
- Required licenses to access SNOMED

**Development Side**

- Eclipse as development environment
- Installed Java Development Kit
- SubVersion server for version control
- GNU/Linux or Windows XP environment
- Internet access for online dictionary support

**End-user Side**

- PostgreSQL Database Management System
- Java Run Time Environment 6
- Windows XP or Recent GNU/Linux Distribution
- Internet Access for online dictionary support

## 4.2. Functional Requirements

### 4.2.1 Text-Mining and Representing Information Formally

RadioRead application will be provided with free-text radiology reports. We have to extract the information in these texts and represent these in a database, in a structured way. We will use Natural Language Processing (NLP) and Machine Learning (ML) techniques for this requirement. NLP requires Morphological, Syntactic and Semantic analysis. We will utilize Zemberek library for morphological analysis, and will use Zargan[7] and TDK[8] online dictionaries in the cases when Zemberek doesn't have the roots of a given word. We will then apply a syntactic analysis step, spotting verbs and noun phrases. For this step we have two choices: first one is to use Link Grammar

Parser, second one is to write our own parser. After this step, we enter the Semantic Analysis step. The verb or verb phrase found in the sentence will be looked up from external dictionaries, such that Zargan, TDK Dictionary, to find synonyms that matches with a predefined “meaning list”. This match (also affected with qualifiers such as “-ma” negativity suffix that inverses meaning of a verb) will be used to mark the information listed in the sentence to be “normal”, “abnormal”, “exists”, “not exists”. The noun phrases found in the Syntactic Analysis step will be broken down to meaningful pieces, and combined with ML methods to find similar records in the database. The information will be recorded in the database, linked with similar records. SNOMED is also considered to be a secondary path for constructing structured information.

### 4.2.2. Statistical Analysis and Information Retrieval

We need to provide reasonable methods to a statistician for querying the accumulated information in the radiology reports. The accumulated information is valuable as a large-scale radiology data mined from free texts, which can be benefited from. A statistician does not only require to query the data using qualifiers mined from the free-texts, but also additional qualifiers (meta information) such as age range, date, frequency.

Besides the needs to analyze accumulated information in a broad sense, a doctor needs to query a specific patient’s history about a specific diagnosis or disease. This way, a doctor can control the progress of a patient without having to search all the reports of the patient for a specific item.

### 4.2.3. Holding Meta Information about Patients and Reports

In order to meet requirements of statistical analysis and information retrieval, we need to store meta information about patients and reports. Meta information of a patient holds fields such as age or gender. We also need to store date, or doctor information with the reports. Besides being useful in statistical analysis, the application has to provide a convenient and intuitive way for users, mostly for doctors, to access data. That’s why we need to hold additional information such as name of a patient.

### 4.2.4. User Interface

- User account management
    - Adding accounts
    - Modifying accounts
- Patient management

    - Adding patients
    - Modifying patient information
    - Listing patients, filtering
- Report management
    - Adding reports associated with patients: This will invoke data mining
    - Listing reports, querying for a patient's specific reports: There needs to be options for querying mined information, besides simple filtering based on meta data
    - Viewing reports
- Statistical querying
    - Query interface: We need a query interface where a user can create his/her queries in an intuitive way, such as constructing free-text like sentences using dropdowns. Our users are not technically skilled, so users need to see the constructed queries in a natural way, for ease of use.

## 4.3. Non-Functional Requirements

We need to provide the user a usable interface, which will require almost no training to learn, and consume minimum time to fill data and query information. Our intended users will be neither skilled nor interested in computers. In order for them to use the application efficiently, the user interface needs to be simple and useful. Especially the statistical query and information retrieval interfaces need to be designed with ease in mind, as these can be complex even for experienced computer users if not designed carefully.

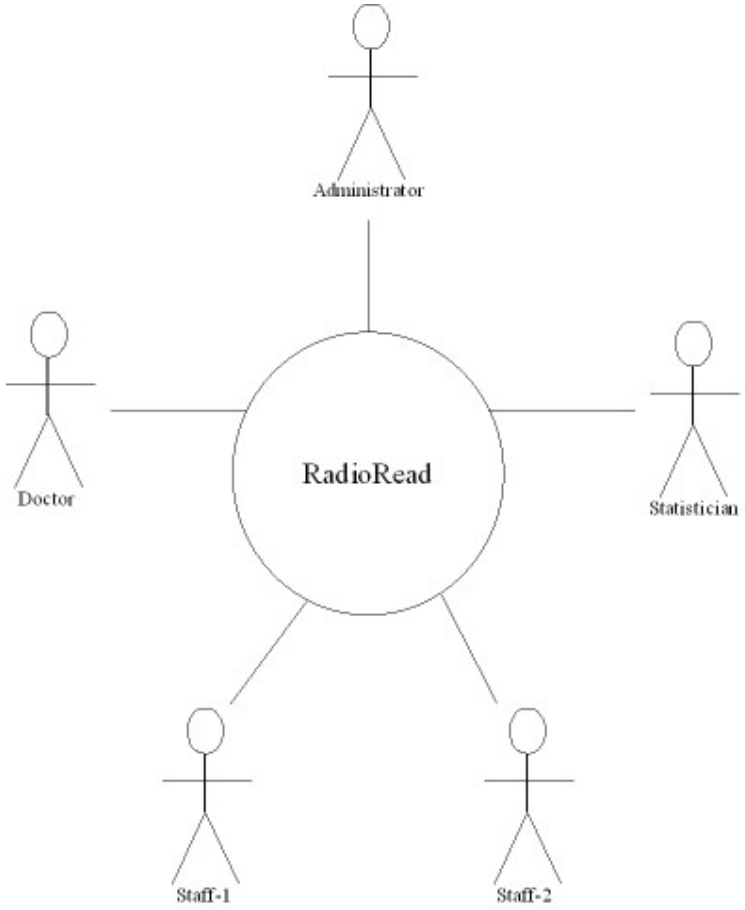
Besides the interface, we need to provide the security of the patients' information. Patients trust doctors and hospitals to store their data, and only the people who are authorized to see their information should be able to view them.

The application needs to be responsive, especially in mining information from reports and querying of mined information. Both reports and queries may be very complicated, but they should not discourage the user because of latency.

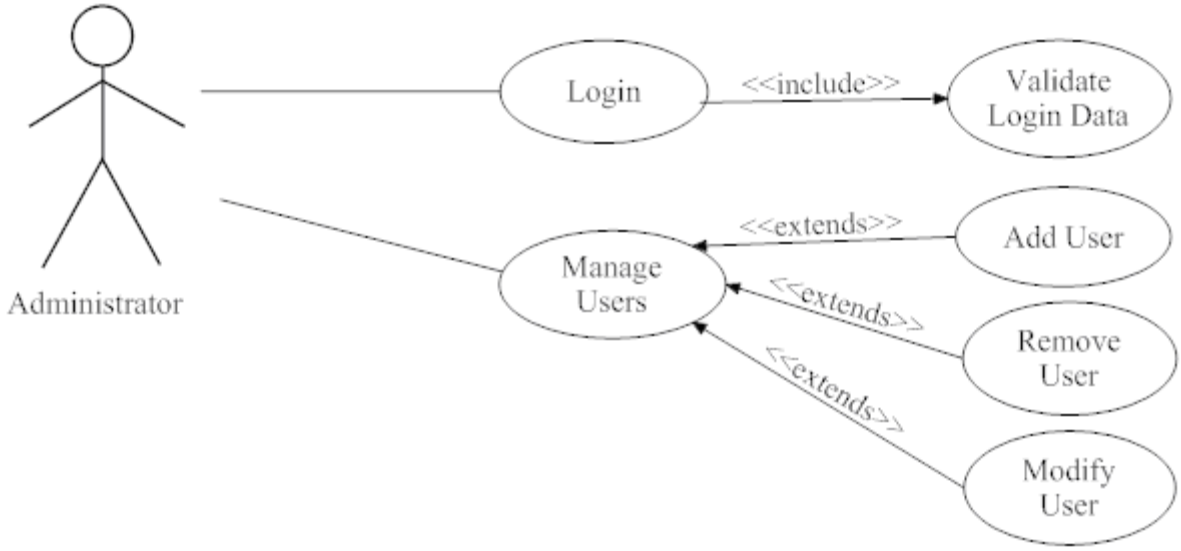
## 4.4. User Requirements

### 4.4.1. Use Case Diagrams

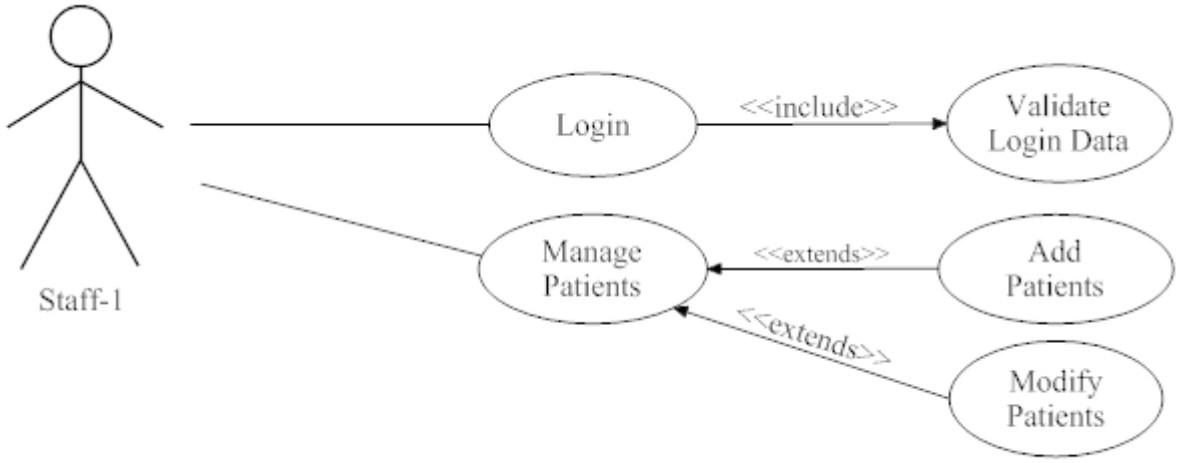
#### 4.4.1.1. Overview



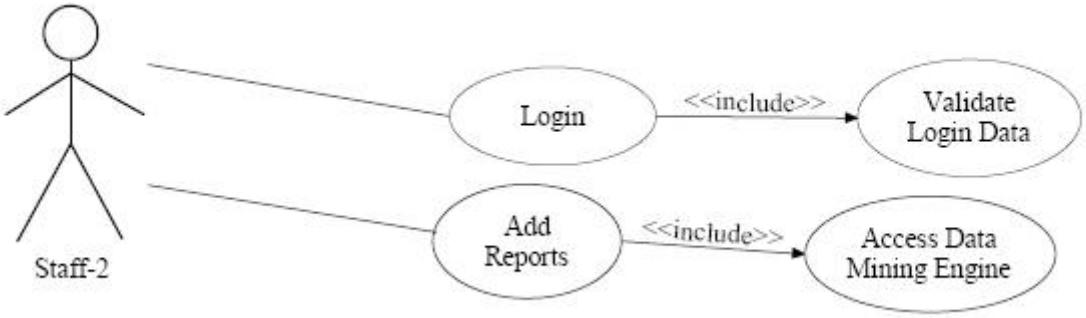

#### 4.4.1.2. Use Case Diagram for Administrator



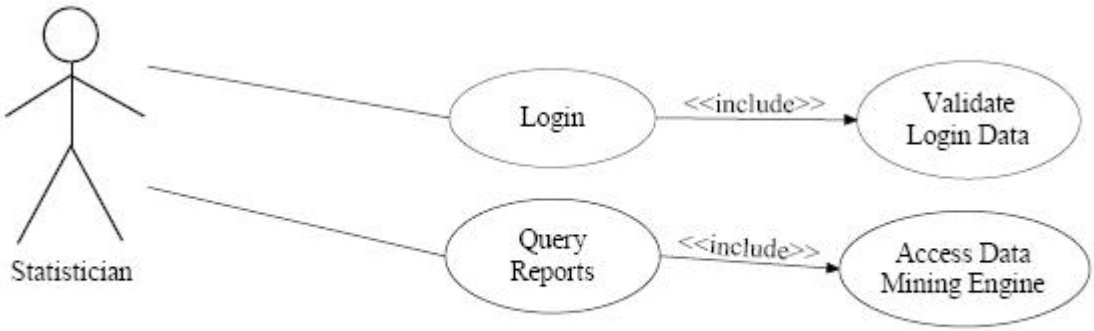
#### 4.4.1.3. Use Case Diagram for Staff-1



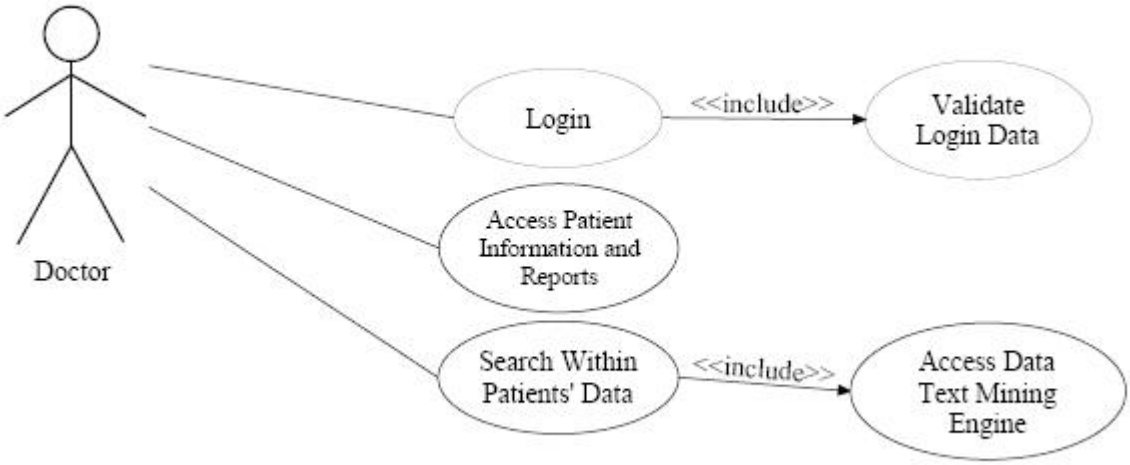

#### 4.4.1.4. Use Case Diagram for Staff-2




#### 4.4.1.5. Use Case Diagram for Statistician



#### 4.4.1.6. Use Case Diagram for Doctor




### 4.4.2. Use Case Scenarios

#### 4.4.2.1. Administrator

- ***Login:*** An administrator has to login to the system in order to realize administrative roles. There will be a user interface for administrative roles. After validation of login information, the administrator will be able to manage users.
- ***Manage Users:*** Administrator may add, remove users and modify the user information. There will be specified user roles and rights and administrator will control users and will be able to restrict the user rights.

#### 4.4.2.2. Staff-1

- ***Login:*** A staff1 has to login to the system in order to realize his/her roles. There will be a user interface for him/her. After validation of login information, the staff1 will be able to manage patients.
- ***Manage Patients:*** Staff1 may add patients and modify the patient information. None of the patients who had been in the clinic will be deleted even if they are dead.

#### 4.4.2.3. Staff-2

- ***Login:*** A staff2 has to login to the system in order to realize his/her roles. There will be a user interface for him/her. After validation of login information, the staff2 will be able to manage reports.

- ***Add Reports:*** Staff2 may add reports to the records of related patients. These reports will then be used for acquiring necessary information.

#### 4.4.2.4. Statistician

- ***Login:*** A statistician has to login to the system in order to realize his/her roles. There will be a user interface for him/her. After validation of login information, the statistician will be able to manage query reports.
- ***Query Reports:*** Statistician may send queries about reports to data mining engine through GUI, and get statistical mined information.

#### 4.4.2.5. Doctor

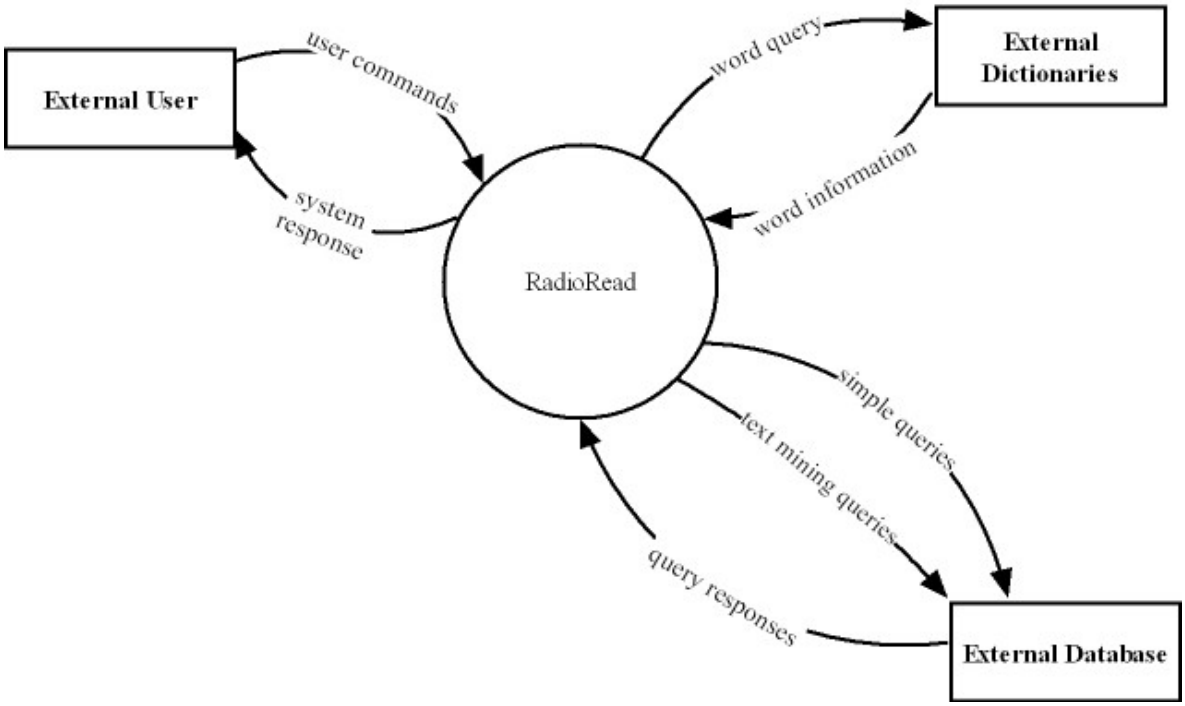
- ***Login:*** A doctor has to login to the system in order to realize his/her roles. There will be a user interface for him/her. After validation of login information, the doctor will be able to manage query reports.
- ***Access Information of Reports:*** Doctor is the only user who can reach the pure text of patients' reports.
- ***Search within Patients Data:*** Doctor may send queries about patients to data mining engine through GUI, and get of mined information of patients.

# 5. MODELLING

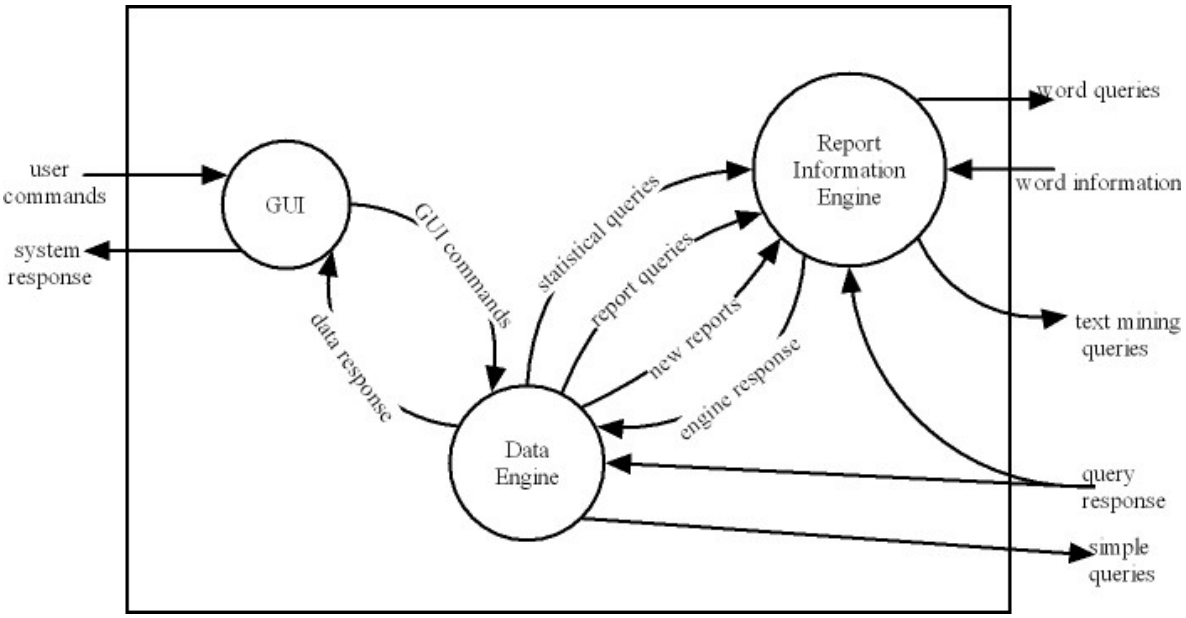
## 5.1. Functional Modelling

### 5.1.1. Data Flow Diagrams

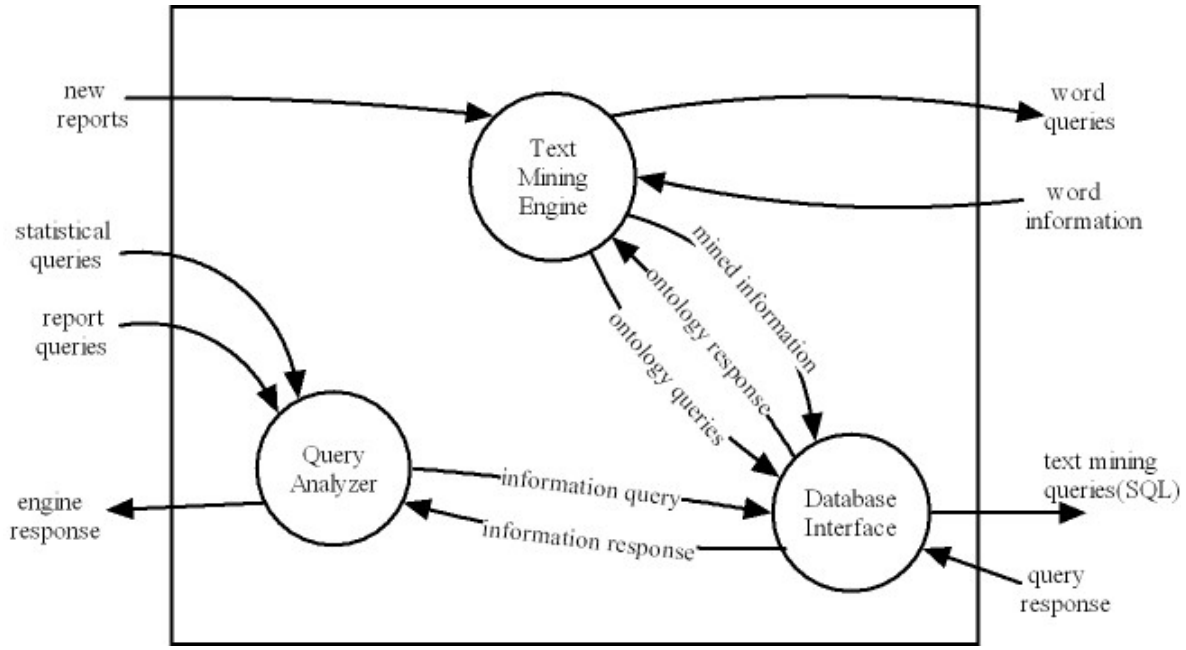
#### 5.1.1.1. Level-0 Data Flow Diagram



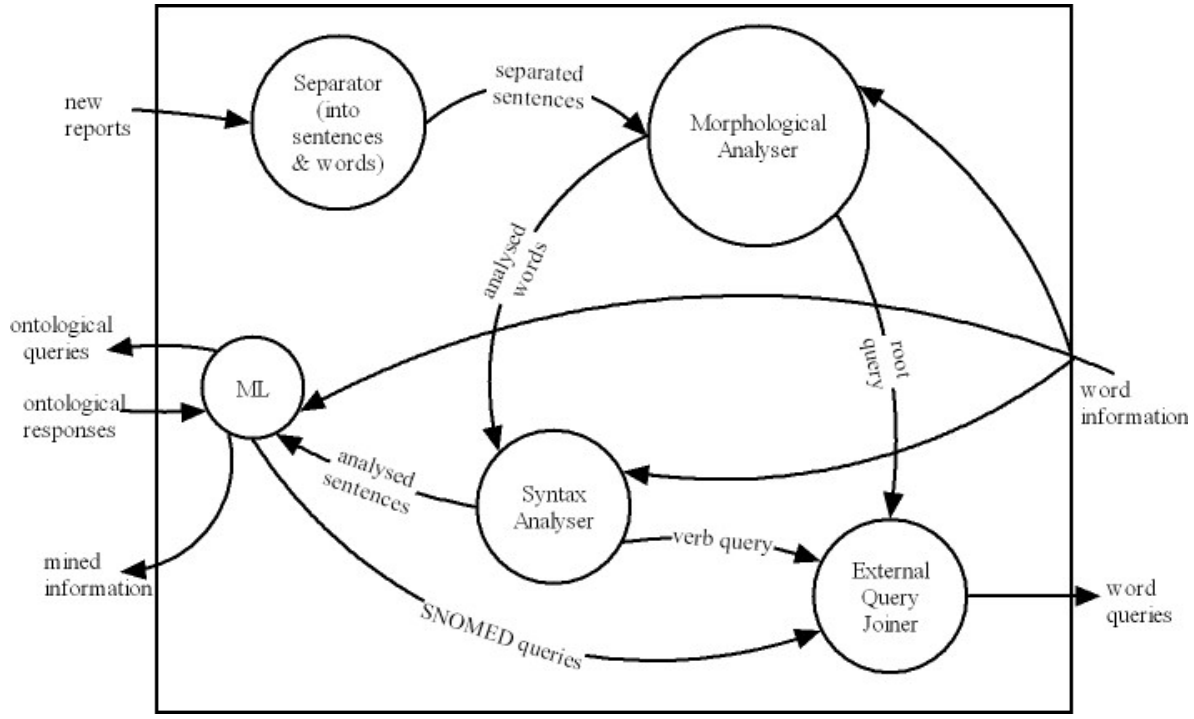
### 5.1.1.2. Level-1 Data Flow Diagram: RadioRead




### 5.1.1.3. Level-2 Data Flow Diagram: Report Information Engine



### 5.1.1.4. Level-3 Data Flow Diagram: Text Mining Engine




## 5.1.2. Data Dictionary

| Name: | Word query |
|---|---|
| Where used? | Output of External Query Joiner (Level 3)<br>Input to External Dictionaries (Level 0) |
| Description | Query that is sent to external dictionaries to get meaning information. |

| Name: | Simple queries |
|---|---|
| Where used? | Output of Data Engine (Level 1)<br>Input to External Database (Level 0) |
| Description | SQL queries that are sent to external database to get/set information which are not mined from reports, but about meta data. |

| Name: | Text mining queries |
|---|---|
| Where used? | Output of Database Interface (Level 2)<br>Input to External Database (Level 0) |
| Description | SQL queries that are sent to external database to get/set information which are mined from reports. |

| Name: | Data Engine |
|---|---|
| Where used? | Level 1 |
| Description | Internal engine that separates data depending on whether they will be sent to Report Information Engine to be text-mined or External Database to be stored. |

| Name: | Report Information Engine |
|---|---|
| Where used? | Level 1 |
| Description | Internal engine that extracts information from reports and handles complex queries such that statistical and report queries. |

| Name: | Statistical queries |
|---|---|
| Where used? | Output of Data Engine (Level 1)<br>Input to Query Analyzer (Level 2) |
| Description | Queries that are about accumulated information |

| Name: | Report queries |
|---|---|
| Where used? | Output of Data Engine (Level 1)<br>Input to Query Analyzer (Level 2) |
| Description | Queries that are about history of a single patient |

| Name: | New Reports |
|---|---|
| Where used? | Output of Data Engine (Level 1)<br>Input to Separator (Level 3) |
| Description | Original text reports that are to be mined. |

| Name: | Text Mining Engine |
|---|---|
| Where used? | Level 2 |
| Description | Internal engine that extracts information from reports by using text mining techniques. |

| Name: | Query Analyzer |
|---|---|
| Where used? | Level 2 |
| Description | Internal analyzer that sends a stream of simpler queries which are obtained from complex queries (statistical/report queries), and merges results. |

| Name: | Information query |
|---|---|
| Where used? | Output of Query Analyzer (Level 2)<br>Input to Database Interface (Level 2) |
| Description | Simpler queries which are obtained from complex queries. |

| Name: | Ontology queries |
|---|---|
| Where used? | Output of ML (Level 3)<br>Input to Database Interface (Level 2) |
| Description | Queries that are sent from Text Mining Engine to database in order to get ontological information (i.e. synonyms, close words…). |

| Name: | Mined information |
|---|---|
| Where used? | Output of ML (Level 3)<br>Input to Database Interface (Level 2) |
| Description | Information obtained from reports using Text Mining Engine is stored in database. |

| Name: | Morphological analyzer |
|---|---|
| Where used? | Level 3 |
| Description | Analyzer that recognizes roots and suffixes of words. |

| Name: | Syntax analyzer |
|---|---|
| Where used? | Level 3 |
| Description | Analyzer that recognizes noun/verb phrases of sentences. |

| Name: | ML |
|---|---|
| Where used? | Level 3 |
| Description | Machine Learning Engine |

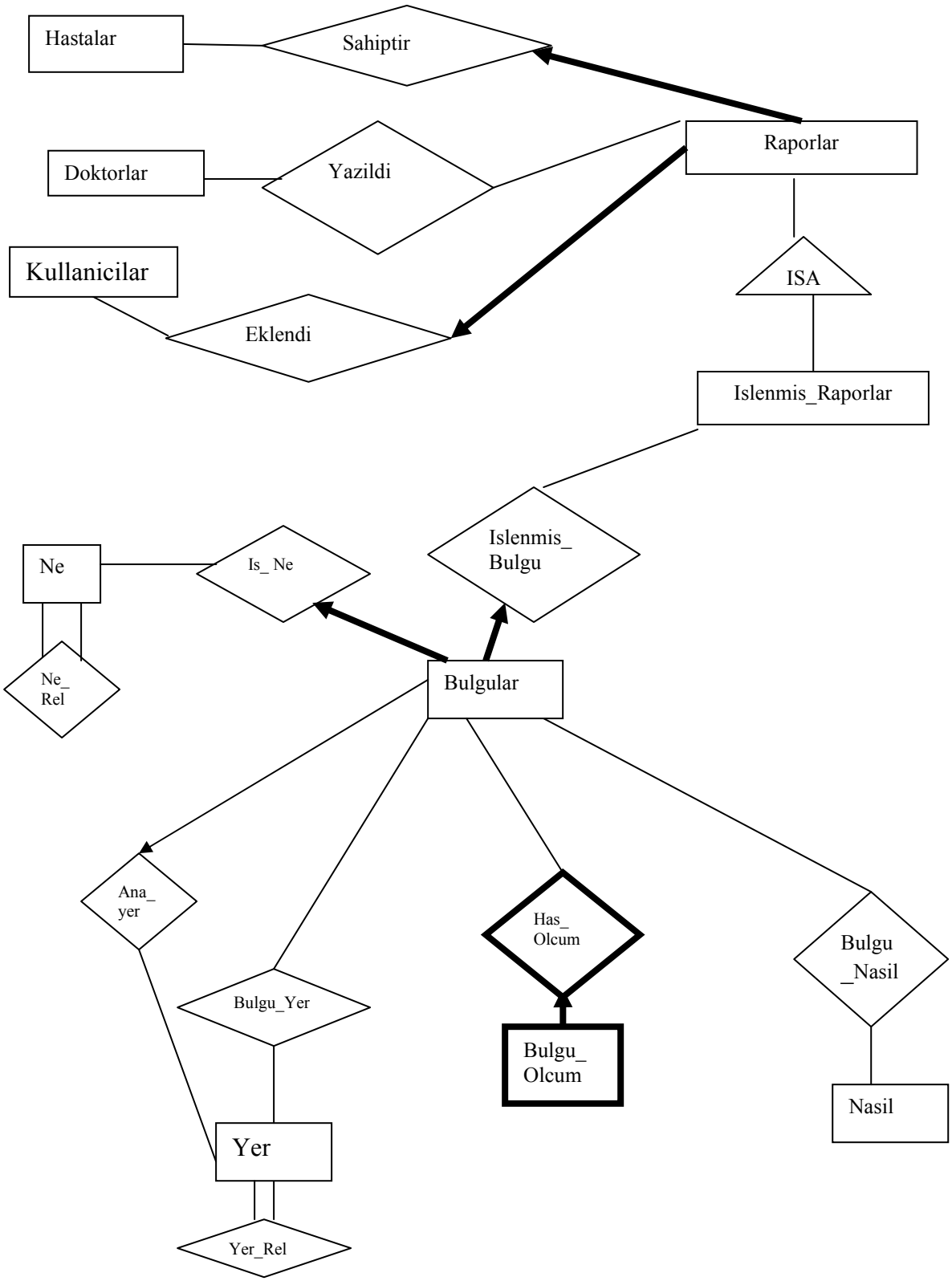
| Name: | Root query |
|---|---|
| Where used? | Output of Morphologic Analyzer (Level 3)<br>Input to External Query Joiner (Level 3) |
| Description | Query of the root information sent to External Dictionaries to learn its meaning via External Query Joiner. |

| Name: | Verb query |
|---|---|
| Where used? | Output of Syntax Analyzer (Level 3)<br>Input to External Query Joiner (Level 3) |
| Description | Query of the verb information sent to External Dictionaries to learn its meaning via External Query Joiner. |

| Name: | SNOMED query |
|---|---|
| Where used? | Output of ML (Level 3)<br>Input to External Query Joiner (Level 3) |
| Description | Query of the specific information of clinical terms sent SNOMED CT via External Query Joiner. |

## 5.2. Data Modelling

### 5.2.1. Entity-Relationship Diagrams



### 5.2.2. Data Descriptions

The data description part gives information about the structure of the database. We have demonstrated entities and relations without their attributes. Instead of this, attributes of entities are listed below. The underlined data are the primary-keys, and the data with stars are foreign keys.

We have 5 global tables that will be populated after doing some text mining on reports. They are “Ne”, “Yer”, “Nasil”, “Yer_Rel” (demonstrating the relation between two “Yer” records) and “Ne_Rel” (demonstrating the relation between two “Ne” records).

**Ne:**

This entity contains information about kinds of all possible findings.

**Yer:**

This entity contains information about locations of all possible findings.

**Nasil:**

This entity contains information about qualities of all possible findings.

**Hastalar:**

This entity contains all necessary information about the patients. This information will be inserted to database through GUI, and they will not be text-mined. This information will be used for diagnostic purposes by doctors.

**Doktorlar:**

This entity contains all necessary information about the doctors that write the reports. This information is gathered from reports. This information will be used for statistical purposes.

**Kullanicilar:**

This entity contains all necessary information about the users. The “Kullanicilar” entity contains information about login information and access-rights. This information will then be used to categorize users into five groups: Admin, Staff-1, Staff-2, Doctor, and Statistician.

**Raporlar:**

This entity contains all necessary information about reports. Each report is owned by a patient, can have multiple doctor information which is written in reports and can be added by only one user. This entity contains only non-mined meta information about reports, such as title, text, date.

**Islenmis_Raporlar:**

This entity is a Raporlar. This entity holds the mined information about reports and separates meta information and mined information.

**Bulgular:**

This entity contains any finding mentioned in the findings ("Bulgular") section of a report text. All information (normal, abnormal, existent, non-existent) that can be extracted from the report text about a single finding is stored here.

**Bulgu_Olcum:**

This entity contains information about quantities of a "Bulgular" record.

**Bulgu_Nasil:**

This relation makes an n-to-n correspondence between "Bulgular" and "Nasil" entitites. This holds qualities of a "Bulgular" record.

**Bulgu_Yer:**

This relation makes an n-to-n correspondence between "Bulgular" and "Yer" entitites. This holds locations of a "Bulgular" record.

**Database Tables:**

Kullanicilar ( user_id, access_rights, username, password, active, name )

Hastalar ( patient_id, name, surname, year_of_birth )

Doktorlar ( doctor_id, title, name, surname )

Raporlar ( report_id, patient_id*, user_id*, title, date, clinical_info, technical_info, diagnosis, findings, result )

Yazildi ( doctor_id*, report_id* )

Islenmis_Raporlar ( report_id*, sure, sure_birimi, normallik )

Bulgular ( bulgu_id, report_id*, ne_id*, yer_id*, normal, var, sonucta_geciyor)

Bulgu_Yer ( bulgu_id*, yer_id*, uzaklik_olcum, uzaklik_birim )

Bulgu_Olcum ( bulgu_olcum_id, bulgu_id*, olcum, olcum_birimi, tur )

Bulgu_Nasil (bulgu_id*, nasil_id*, sonuctan )

Yer ( yer_id, isim )

Yer_Rel ( birincil_yer_id*, ikincil_yer_id* )

Nasil ( nasil_id, isim )

Ne ( ne_id, isim )

Ne_Rel ( birincil_ne_id*, ikincil_ne_id* )

| **Raporlar** | |
|---|---|
| report_id* | |
| patient_id* | |
| user_id* | |
| title | The title text of the report |
| date | Date of the report |
| clinical_info | The text in the Clinical Information section |
| technical_info | The text in the Technical Information section |
| findings | The text in the Findings section |
| result | The text in the Results section |

| **Islenmis_Raporlar** | |
|---|---|
| report_id* | |
| sure | Quantity of the advised time for next consultation. Can be NULL if not specified in the Results section of the report |
| sure_birimi | Unit of the time |
| normallik | True / False / NULL – Holds whether normality / abnormality is specified in the Results section of the report |

| **Bulgular** | |
|---|---|
| bulgu_id | |
| report_id* | |
| ne_id* | |
| yer_id* | Holds the primary location of this finding. Bulgu_Yer table holds secondary locations |
| normal | True / False / NULL – holds whether this finding is specified as normal or abnormal or not specified in normality |
| var | True / False / NULL – holds whether this finding is specified as existent or non-existent or not specified in existence |
| sonucta_geciyor | True / False – Holds whether this finding is also referenced in the results section of the report |

| **Bulgu_Yer** | |
|---|---|
| bulgu_id* | |
| yer_id* | |
| uzaklik_olcum | The quantity of the distance |
| uzaklik_birim | The unit of the measurement |

| **Bulgu_Olcum** | |
|---|---|
| bulgu_olcum_id | |
| bulgu_id* | |
| olcum | The quantity of the measurement |
| olcum_birimi | The unit of the measurement |
| tur | The kind of the measurement (i.e. “çap”, “hız”, “uzunluk”, “boyut”) |

| **Bulgu_Nasil** | |
|---|---|
| bulgu_id* | |
| nasil_id* | |
| sonuctan | True/False. Holds whether this quality is gained only from the “Results” section of a report |

| **Yer** | |
|---|---|
| yer_id | |
| isim | Name of the quality (i.e “meme”, “sol meme”, “areola” ) |

| **Yer_Rel** | |
|---|---|
| birincil_yer_id* | |
| ikincil_yer_id* | |

| **Nasil** | |
|---|---|
| nasil_id | |
| isim | Name of the quality (i.e “dens”, “heterojen”, “solid (lezyon)” ) |

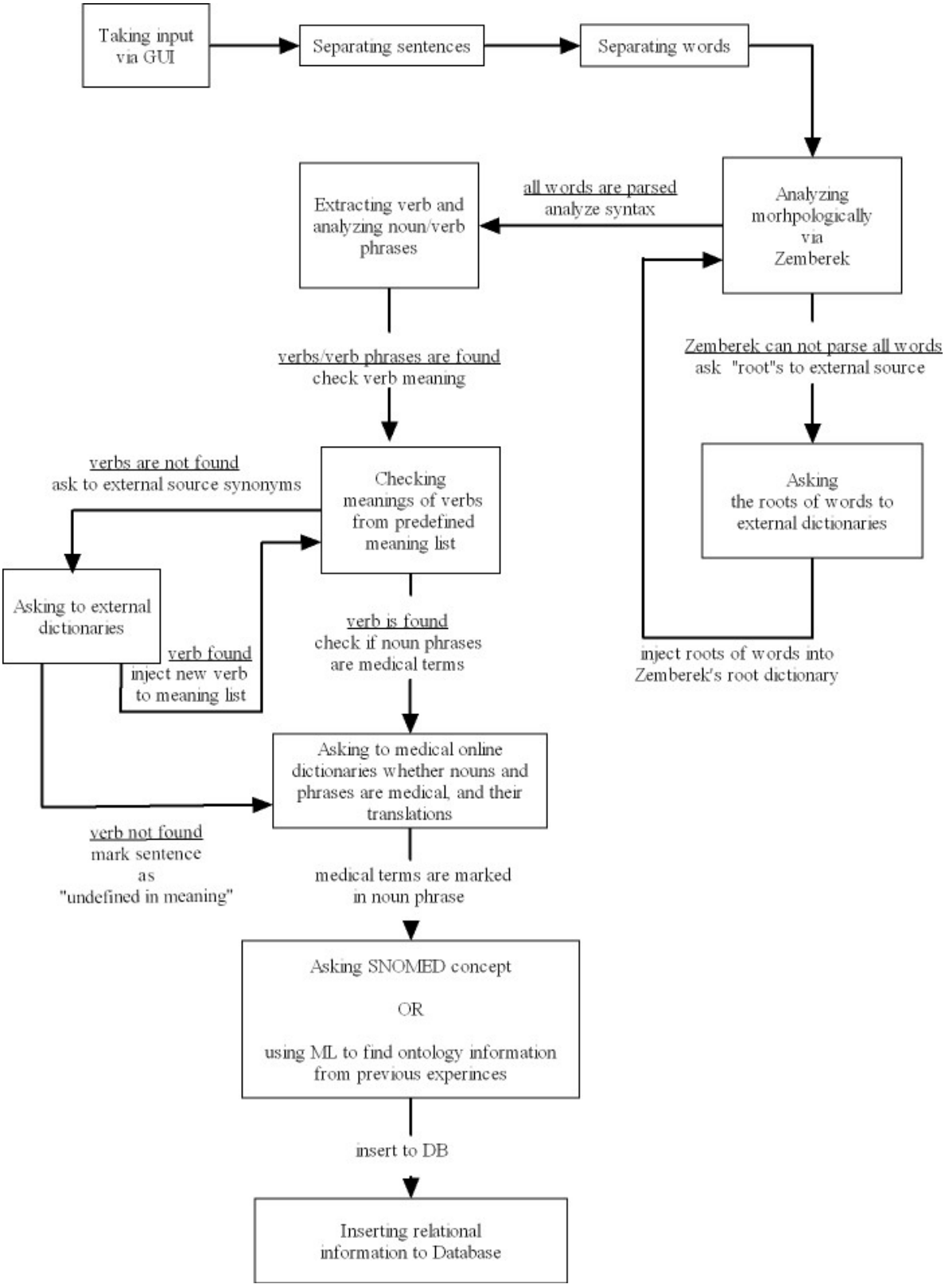
| **Ne** | |
|---|---|
| Ne_id | |
| isim | Name of the finding (i.e “duktal ektazi”, “patern”, “lezyon”) |

| **Ne_Rel** | |
|---|---|
| birincil_ne_id* | |
| ikincil_ne_id* | |

## 5.3. Behavioral Modelling

### 5.3.1. State Transition Diagram

Our main aim is to extract information from radiology reports using NLP techniques. When a report is entered to the system, it will be separated into sentences and the sentences will be separated into words. These words will be morphologically analyzed after separation process via Zemberek. If Zemberek can not parse the words, the program asks the words to an external dictionary and gets root information. The root information taken from the external dictionary will be injected to Zemberek's root database. After the morphological analysis, syntactic analysis state starts. In this state, constituents of the sentence, especially noun and verb phrases, are extracted. For this state we have two choices: first one is to use Link Grammar Parser, second one is to write our own parser. After extracting verb or verb phrase from the sentence, their meaning is looked up from external dictionaries, such as Zargan or TDK Dictionary. If the verbs' meaning can be found in this way they will be added to an internal dictionary, otherwise the sentence will be marked as "undefined in meaning". After this step, the nouns and noun phrases are looked up from external dictionaries to find out whether they are medical terms or not. The medical terms that are found in this step are sent to ML State for further information. ML methods and/or querying SNOMED are considered to be used as the last steps to construct structured information that is suitable to be stored in a database.


Taking input via GUI
Separating sentences
Separating words
Analyzing morhpologically via Zemberek
all words are parsed
analyze syntax
Extracting verb and analyzing noun/verb phrases
Zemberek can not parse all words
ask "root"s to external source
Asking the roots of words to external dictionaries
inject roots of words into Zemberek's root dictionary
verbs/verb phrases are found
check verb meaning
verbs are not found
ask to external source synonyms
Checking meanings of verbs from predefined meaning list
Asking to external dictionaries
verb found
inject new verb to meaning list
verb is found
check if noun phrases are medical terms
Asking to medical online dictionaries whether nouns and phrases are medical, and their translations
verb not found
mark sentence as "undefined in meaning"
medical terms are marked in noun phrase
Asking SNOMED concept
OR
using ML to find ontology information from previous experinces
insert to DB
Inserting relational information to Database

# 6. PROJECT ESTIMATIONS

Estimations are essential to have a general idea about the schedule, cost and effort. These are required in the early phases of the project. Then we restore the estimates with the help of metrics. We will use these metrics to determine progress and to estimate future projects. If we can make our plans according to these estimations then it will be easier to manage risks and increase efficiency.

***Function Point Estimation***

| | **#simple** | **#average** | **#complex** | **simple** | **average** | **complex** | |
|---|---|---|---|---|---|---|---|
| **number of user inputs** | *6* | *3* | *5* | 3 | 4 | 6 | 60 |
| **number of user outputs** | *2* | *7* | *2* | 4 | 5 | 7 | 57 |
| **number of inquiries** | *4* | *2* | *2* | 3 | 4 | 6 | 32 |
| **number of files** | *20* | *10* | *5* | 7 | 10 | 15 | 315 |
| **number of external interfaces** | *1* | *4* | *2* | 5 | 7 | 10 | 53 |
| | | | | | | ***Total Count*** | ***517*** |

*1. Does the system require reliable backup and recovery?*

Answer: Yes, 4.

*2. Are data communications required?*

Answer: Yes, 4.

*3. Are there distributed processing functions?*

Answer: Yes, 2.

*4. Is performance critical?*

Answer: Yes, 3.

*5. Will the system run in an existing, heavily utilized operational environment?*

Answer: Yes, 2.

*6. Does the system require on-line data entry?*

Answer: Yes, 3.

*7. Does the on-line data entry require the input transactions to be built over multiple screens or operations?*

Answer: Yes, 2.

*8. Are master-files updated online?*

Answer: No, 0.

*9. Is the internal processing complex?*

Answer: Yes, 5.

*10. Is the code designed to be reusable?*

Answer: Yes, 4.

*11. Are inputs, outputs, files, or inquiries complex?*

Answer: Yes, 5.

*12. Are conversion and installation included the design?*

Answer: No, 0.

*13. Is the system designed for multiple installations in different organizations?*

Answer: Yes, 1.

*14. Is the application designed to facilitate change and ease of use by the user?*

Answer: Yes, 4.

```
FP(Function Point) = count-total x [ 0.65 + 0.01 x ΣFi]
= 517 x [0.65 + 0.01 x 39]
= 538

LOC = FP x 30 (30 since we will use Java)
LOC = 16,140
```

***COCOMO (Basic)***

| **Software Project** | $a_b$ | $b_b$ | $c_b$ | $d_b$ |
|---|---|---|---|---|
| Organic | *2.4* | *1.05* | *2.5* | *0.38* |
| Semi-detached | *3* | *1.12* | *2.5* | *0.35* |
| Embedded | *3.6* | *1:2* | *2.5* | *0.32* |

We decided that our project is semi-detached.

$E = a_b \times ( KLOC )^{b_b}$
$E = 3.0 \times ( 16.14 )^{1.12}$
**E = 67 person-months**

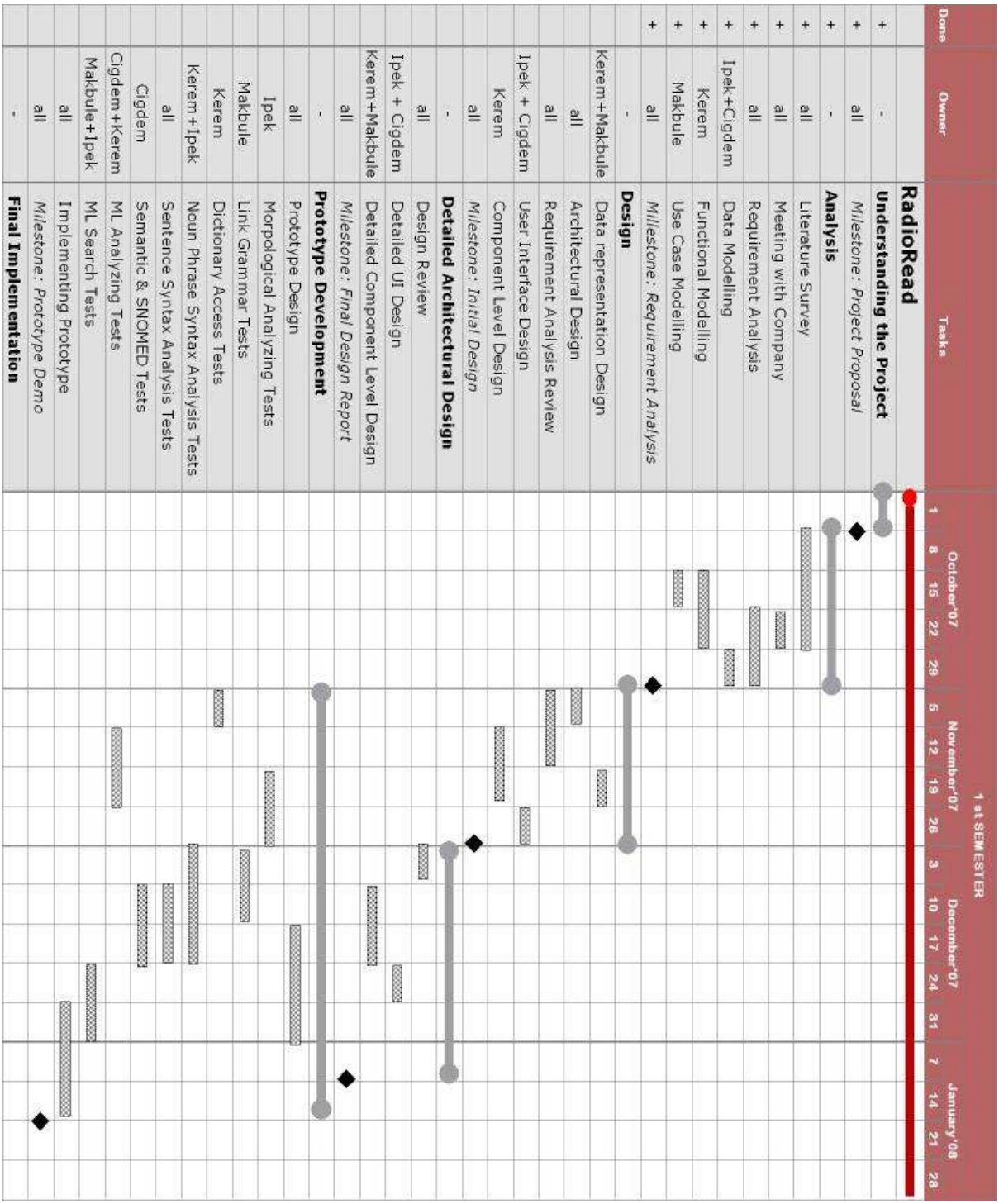
$D = c_b \times ( E )^{d_b}$
$D = 2.5 \times 67^{0.35}$
**D = 11 months**

P = E / D
P = 67 / 11
**P = 6 persons**

# 7. RISK MANAGEMENT

| Risk | Probability | Impact | Risk Mitigation Monitoring and Management |
|---|---|---|---|
| Lack of required skills | 20% | 4 | Increase the workloads of team members to acquire these skills in time. |
| Unexpected overloads | 70% | 4 | Manage time well and increase work. |
| Poor time management | 50% | 4 | Observe the improvement of the project every week trying to obey the schedule. Accordingly, work harder or contact the instructor and drop some features of the final deliverable product if acceptable by the instructor. |
| Temporarily unavailability of one or more of the team members | 5% | 3 | Increase the workloads of other team members. |
| Severance of a team member | 5% | 3 | Increase the workloads of other team members. |
| Lack of knowledge about application area | 60% | 4 | Study application area thoroughly. |
| Polarization between team members | 10% | 4 | Solve the problem anyway. |
| Physiological problems | 40% | 2 | Show more tolerance to that member. |
| Irresponsibility of members | 15% | 4 | Show no tolerance. Give some reasonable punishment. |
| Missing the meetings twice. | 10% | 2 | Give some reasonable punishment. |
| Missing deadlines. | 50% | 4 | Increase the workloads of team members. |
| Treating rude and being offended to other members | 10% | 4 | Behave as nothing has happened during the team work & meetings. |

# 8. PROJECT SCHEDULE



| Done | Owner | Tasks | 2nd SEMESTER<br>February'08 | March'08 | April'08 | May'08 |
|---|---|---|---|---|---|---|
| | | **RadioRead** | |
| + | - | **Understanding the Project** | |
| + | all | *Milestone: Project Proposal* | |
| + | - | **Analysis** | |
| + | all | Literature Survey | |
| + | all | Meeting with Company | |
| + | all | Requirement Analysis | |
| + | Ipek+Cigdem | Data Modelling | |
| + | Kerem | Functional Modelling | |
| + | Makbule | Use Case Modelling | |
| + | all | *Millestone: Requirement Analysis* | |
| | - | **Design** | |
| | Kerem+Makbule | Data representation Design | |
| | all | Architectural Design | |
| | all | Requirement Analysis Review | |
| | Ipek + Cigdem | User Interface Design | |
| | Kerem | Component Level Design | |
| | all | *Milestone: Initial Design* | |
| | - | **Detailed Architectural Design** | |
| | all | Design Review | |
| | Ipek + Cigdem | Detailed UI Design | |
| | Kerem+Makbule | Detailed Component Level Design | |
| | all | *Milestone: Final Design Report* | |
| | - | **Prototype Development** | |
| | all | Prototype Design | |
| | Ipek | Morpological Analyzing Tests | |
| | Makbule | Link Grammar Tests | |
| | Kerem | Dictionary Access Tests | |
| | Kerem+Ipek | Noun Phrase Syntax Analysis Tests | |
| | all | Sentence Syntax Analysis Tests | |
| | Cigdem | Semantic & SNOMED Tests | |
| | Cigdem+Kerem | ML Analyzing Tests | |
| | Makbule+Ipek | ML Search Tests | |
| | all | Implementing Prototype | |
| | all | *Milestone: Prototype Demo* | |
| | - | **Final Implementation** | |

# 9. CONCLUSION

This report includes the general aspects of our project and is also a guide for the reader to get the general idea of the project. During the preparation of this report, we have gained insight for our project. Some points that seem complex at the beginning of the term are clear for the team after this report. Our project is scheduled to spend our effort more efficiently during all semester. Also we listed our general requirements to determine our basic functionalities and drew diagrams to make the implementation easier.

The process, from the beginning to the end, will be heavily-loaded and challenging, but we believe in the success of our team and our project. Our users will easily realize the difference of RadioRead when it takes its place in the market.

# 10. REFERENCES


[1] Zemberek Library, *http://zemberek.googlecode.com*

[2] SNOMED-CT, *http://www.snomed.org*

[3] Parsing English with a Link Grammar (Daniel D. K. Sleator, Davy Temperley, October 1991), *http://arxiv.org/PS_cache/cmp-lg/pdf/9508/9508004v1.pdf*

[4] Link Grammar, *http://www.link.cs.cmu.edu/link/*

[5] A Link Grammar for Turkish (Özlem İstek, August 2006), *http://www.cs.bilkent.edu.tr/~ilyas/PDF/THESES/ozlem_istek_thesis.pdf*

[6] CLEF (Clinical e-Science Framework), *http://www.clinical-escience.org*

[7] Zargan English Turkish Online Dictionary, with Roche Medical Dictionary, *http://www.zargan.com*

[8] TDK (Türk Dil Kurumu) Online Dictionary, *http://www.tdk.gov.tr*