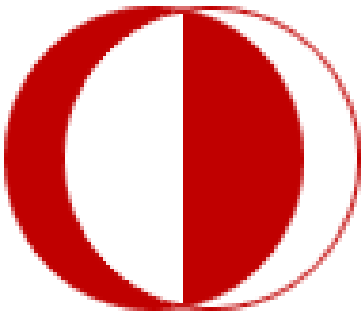
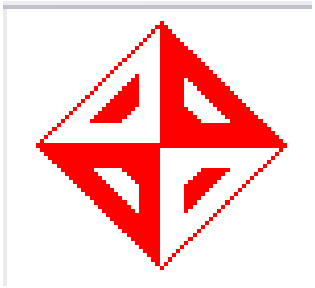


MIDDLE EAST TECHNICAL UNIVERSITY



COMPUTER ENGINEERING DEPARTMENT

SENIOR PROJECT 2007

REQUIREMENT ANALYSIS REPORT

ONUR AK 1394576

ŞERİF ÇETİNER 1394832

MASHAR TEKİN 1395565

SADETTİN ŞEN 1395540

INDEX

1. INTRODUCTION

The general idea of text mining – getting small "nuggets" of desired information out of "mountains" of textual data without having to read it all – is nearly as old as information retrieval (IR) itself. Currently text mining is enjoying a surge of interest fuelled by the popularity of the Internet, the success of bioinformatics, and a rebirth of computational linguistics. It can be viewed as one of a class of non-traditional IR strategies which attempt to treat entire text collections holistically, avoid the bias of human queries, objectify the IR process with principled algorithms, and "let the data speak for itself."

Text mining is data mining applied to textual data. Text is "unstructured, amorphous, and difficult to deal with" but also "the most common vehicle for formal exchange of information." Therefore, the "motivation for trying to extract information from it is compelling – even if success is only partial.

'To the working scientist or engineer, time spent gathering information or writing reports is often regarded as a wasteful encroachment on time that would otherwise be spent producing results that he believes to be new' [Weinberg et al, 1963] …. The intelligence analyst, by contrast, is much more intimate with the available base of recorded information. New knowledge, or finished intelligence, is seen as emerging from large numbers of individually unimportant but carefully hoarded fragments that were not necessarily recognized as related to one another at the time they were acquired. Use of stored data is intensively interactive; "information retrieval" is an inadequate and even misleading metaphor. '
Don R. Swanson (1988)

A true text mining system

- must operate on ***large, natural language text*** collections.
- must use ***principled algorithms*** more than heuristics and manual filtering.
- must extract ***phenomenological units of information*** (e.g., patterns) rather than or in addition to documents.
- must discover ***new knowledge***.

In a text mining system, **machine learning** must be used. Mjolsness and DeCoste (2001) defines machine learning as *"The study of computer algorithms capable of learning to improve their performance of a task on the basis of their own previous experience primarily through pattern recognition and statistical inference."*.

Natural language processing (NLP) is a subfield of artificial intelligence and computational linguistics. It studies the problems of automated generation and understanding of natural human languages. Natural language generation systems convert information from computer databases into normal-sounding human language, and natural language understanding systems convert samples of human language into more formal representations that are easier for computer programs to manipulate.

2. PROBLEM DEFINITION

Radiology reports contain a great deal of information that characterizes a patient's medical condition. However, a large percentage of this information is unstructured, taking the form of free text, and is therefore difficult to search, sort, analyze, summarize, and present. Structured medical data have so much potential benefits for medical practice, research, and teaching. For example, if a radiologist is interested only in a given clinical episode, he or she may elect to retrieve only those **reports** in which the relevant anatomy or findings are described. Accurate extraction of lesion size information from **radiology reports** can allow a system to automatically construct a growth timeline for an indicator lesion. For research and teaching, structured **reports** can greatly improve the recall and precision of information retrieval tasks.
Especially, radiological education can be speed up by the presentation and instruction of pathologically different and similar images of the same disease. Because of this, the presence of an image achieve is crucial for radiological education and a high quality radiology service. The retrieval of similar patient and diseases after a free text search of archieved reports to radiology workers and/or trainees is going to supply efficiency and convenience for diagnosis and interpretation. It's been contemplated that free-text seach capability shall be applicable to all medical reports and shall be widely used within an hospital, independent of radiological education and service quality increase. This will lead way to numerous analysis such as diagnosis categorization, identification of treatment integrity and variances for patients who had the same diagnosis and audit of test adequacy.

Only structured data are amenable to advanced causal, spatial, temporal, and evolutionary database modeling techniques that are now being developed in the fields of medical informatics and computer science. The implications for teaching files and data collection for retrospective research studies are obvious.

However, automatic structuring of **radiology reports** is a difficult task for the following reasons:

- Automatic structuring requires deep understanding because it is desirable to translate all relevant information into structured form.
- Automatic structuring must deal with ungrammatical writing styles. Shorthand and telegraphic writing styles are comemon in **radiology reports**.
- The vocabulary is large. Large numbers of complex medical terms, proper names, product names, abbreviations, and staging codes are used in **radiology reports**. Hundreds of descriptive adjectives are used that are not found in any common electronic medical glossaries.
- There is an assumed knowledge between the writer and reader.

The objective of this project is text-mining on free-text(unstructured) radiology reports to get meaningful information, storing these in the database and integrating with the existing information system to make the patient records searchable.

Natural language processing methods have been reported to increase success in this subject, that is known as text-mining. Very limited theoretical research exists for Turkish language and yet no applied work exists. And also there are no open-source natural language processing tools for Turkish. In addition to this, what information is going to extracted and what information is important is not yet determined. This projected system is going to self-train itself on these.

3. LITERATURE SURVEY

3.1.Tokenizer

For the language Turkish, we have a morphological tokenizer which is written by Kemal Oflazer but we want a sentence tokenizer for our project. From the researches i have made in the internet, i have found an article which will be helpful for developing sentence boundary detection. From the article, which we can be found at
(http://www.linguistics.ruhr-uni-bochum.de/~strunk/ks2005FINAL.pdf)
Turkish is mostly a verb-final language. So identifying a verb which is followed by a full-stop means the end of a sentence in Turkish most of the time. But to be more accurate they have implemented some other criterions.

They trained a program in an unsupervised method manner with the METU TURKISH CORPUS. They have seen that there are also abbreviations and the ratio of this in the corpus for Turkish language was 2.84% and they got a 1.31% error for detecting the sentence boundaries.

Another approach was proposed by Özlem Aktaş who is from Dokuz Eylül University.
(http://ab.org.tr/ab06/bildiri/68.doc) In this approach, every sentence must end with some specific punctuations. We combine this knowledge with a rule list and a abbreviation list. You can see the rules from the document which can be found at the upper html address. By using these 2 lists we can find the sentence boundaries.

So we can use an unsupervised learning method with a corpus and by the help of a Turkish morphological analyser. Maybe we can simply say that the full-stop seperates two sentences but this can be a problem with abbreviations. And we can use the second method. The important thing here is that how our datum are written meaning the syntax and how will the methods will perform over the datum of our project. By looking the performance we will decide the method.

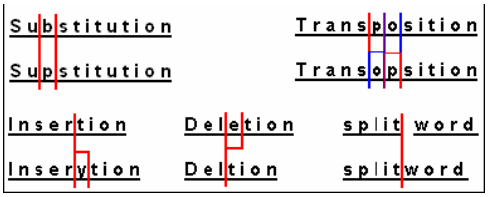
3.2.Spell-Checking

We want to be able to correct miswritten words so to get a higher valuable data at the end of our Project so we have decided to have a spell-checker in our program which corrects the miswritten words. From the articles i have found we can make use of statistics of the syllables. The article shows this method for this
(http://ab.org.tr/Yazismalar/2006/sunum/att-0251/01-E00-373161051.pdf)

It has been seen that Turkish language has at most 5 letter syllable. So we seperate the words into the syllables and then we have the percentages of the syllables' used. From these statistics and with the help of some more restrictions we can seperate the input into their syllables' and then by looking the statistics we can correct the miswritten words. It has been told that this method has nearly 100% success in correcting the miswritten words.

Another article uses dictation errors which is language independent.
(http://www.icgst.com/AIML05/papers/P1120535133.pdf)
They look for 5 commonly mistyping errors which are :

- **Substitution Error:** Using a letter instead of the other.
- **Deletion Error:** Unintended elimination of one or more letters.
- **Insertion Error:** Unintended insertion of a letter in a word.
- **Transposition Error:** Transposition of two adjacent letters.
- **Split Word Error:** Attaching two correct separate words.



Again we need a corpus to train this algorithm.
We can use both of these algorithms or a combination of them in our project. But the first one is giving nearly 100% success as been told and if it is so , we can try to do that method.

3.3.Machine Learning

For the machine learning method, we have not decided which one we will continue with. But we have found that for text mining or data mining needs statistical information and since our inputs are of the same type we will use these statistics with a supervised learning method. From the inputs to our program we will be able to separate the disease names , which are mostly Latin words then we will keep track of the statistics of these words. We have ended up with this idea by looking at some articles from the internet. For example, our project is similar with e-mail filtering, and from the article
(http://people.csail.mit.edu/jrennie/papers/ifile00.pdf) we can see that statistics is used in this project. But the class of the machine learning method is not decided yet.

3.4. Morphological Analysis

Morphology is the field within linguistics which studies the internal structure of words. While the smallest units of syntax are generally known as words, it is clear that in most (if not all) languages, words can be related to other words by rules. For example, English speakers recognize that the words *dog*, *dogs*, and *dog-catcher* are closely related. These relations are recognized from English speakers' tacit knowledge of the rules of word-formation in English. They intuit that *dog* is to *dogs* as *cat* is to *cats*; similarly, *dog* is to *dog-catcher* as *dish* is to *dishwasher*. The rules understood by the speaker reflect specific patterns in the way words are formed from smaller units and how those smaller units interact in speech. In this way, morphology is the branch of linguistics that studies patterns of word-formation within and across languages, and attempts to formulate rules that model the knowledge of the speakers of those languages.

There are three principal approaches to morphology, which each try to capture the distinctions above in different ways and are efficient for different languages. These are,

- Morpheme-based morphology, which makes use of an Item-and-Arrangement approach.
- Lexeme-based morphology, which normally makes use of an Item-and-Process approach.
- Word-based morphology, which normally makes use of a Word-and-Paradigm approach.

In morpheme-based morphology, word-forms are analyzed as arrangements of morphemes. The minimal meaningful unit of a language is defined as **morpheme**. In a word like *independently*, the morphemes are *in-*, *depend*, *-ent*, and *ly*; *depend* is the root and the other morphemes are, in this case, derivational affixes. In a word like *dogs*, we say that *dog* is the root, and that *-s* is an inflectional morpheme. This way of analyzing word-forms as if they were made of morphemes put after each other like beads on a string, is called Item-and-Arrangement. Since Turkish is an agglutinative language and has word structures that are formed by productive affixations of derivational and inflectional suffixes to root words, we will use the morpheme-based morphology for our Morphological Analyzing process. To see Turkish word formation as an example is:

kesilemedi
And its morphemes are:
kes +il +eme +di
As it is seen a word can be formed by a root and lots of suffixes. As a result, although Turkish has finite-state but nevertheless rather complex morphomatics, which makes it difficult to analyzed and processing. Adding morphemes to a root word or a stem, a word can be converted from a nominal to a verbal structure or vice-versa. The surface realizations of morphological constructions are constrained and modified by a number of phonetic rules. Although a small number of exceptional cases are available, vowels in the affixed morpheme must agree with the preceding vowel in certain aspects to achieve vowel harmony. Under certain circumstances, vowels in the roots and morphemes are deleted. Also, the assimilation of a large number of words into the language from various foreign languages -French, Arabic and Persian – have resulted in word formations and these words behave as exceptions to many rules. Because of these difficulties and the inadequacy of researches there is not a complete Turkish language processing tool. However, there are some software's currently used, with their demo versions.
The project currently used for Turkish language processing tool is 'zemberek' which is an open source project. Zemberek can break text into its tokens, and these tokens into their morphemes. After this, it can accomplish the Part Of Speech (POS) tagging. Although in our input texts there are lots of words that are not Turkish, probably they are Latin or English, it can be used in pre-processing. Since this is an open project we can use the necessary parts for our texts after doing some modifications. These modifications will include a module that process the Medical terms.

http://en.wikipedia.org/wiki/Morphology_(linguistics)
http://citeseer.ist.psu.edu/cache/papers/cs/129/ftp:zSzzSzftp.cs.bilkent.edu.trzSzpubzSztech-reportszSz1994zSzBU-CEIS-9423.pdf/using-a-corpus-for.pdf

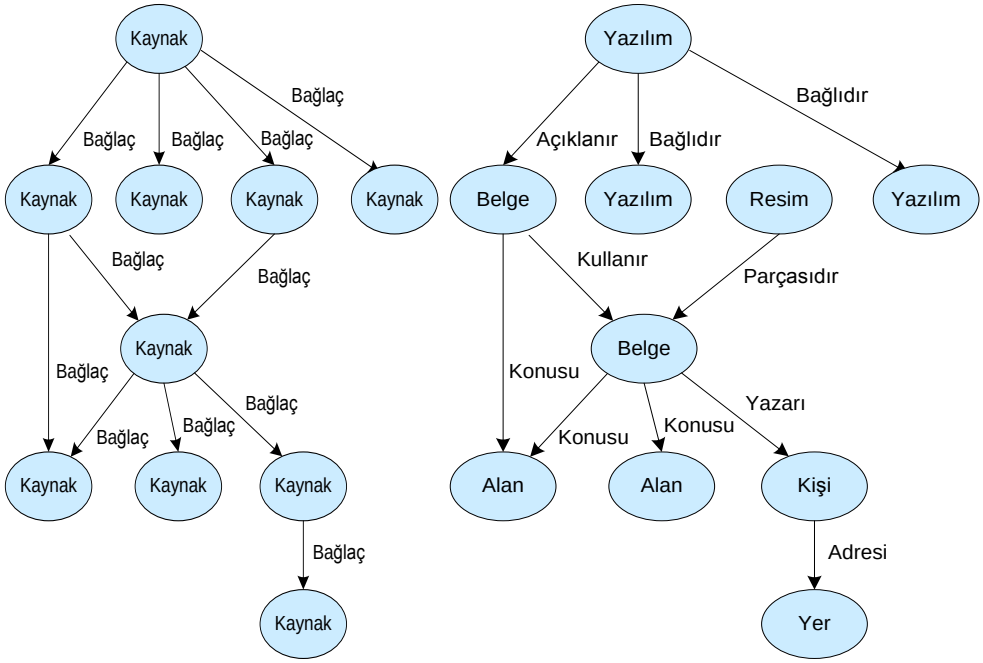
3.5. Ontology

While we are brain storming on the project, we realize that we need something in order to provide us relations between words after morphology phase. Searching on that, we learned that we need ontology on that aspect.
There is lot of definitions of ontology. However, one of these helps us to understand what we can do with ontology. That definition belongs to Fredrik Arvidsson and Annika Flycht-Eriksson. They said that "An ontology provide a shared vocabulary, which can be used to model a domain, that is, the type of objects and/or concepts that exist, and their properties and relations from general to specific

- Generic
- Core
- Domain
- Task
- Application"

That document also stated that Philosophy, Library and Information Science, Artificial Intelligence, Natural Language Processing, The Semantic Web are the fields that needs ontology. (http://www.ida.liu.se/~janma/SemWeb/Slides/ontologies1.pdf)

The Semantic Web example is the most popular one nowadays. The diagram prepared by Doç. Dr. Selim Akyokuş *(Bilgisayar Mühendisliği Bölümü, Doğuş Üniversitesi)* shows the mission of ontology clearly below:



WEB TODAY SEMANTIC WEB

As we can understand by semantic web computers can understand what they have and what they can deliver. Ontology is the way that creates the relations. That we need in our project. We try to reach meaningful relations between words in order to determine what the illness is or whether the patient healthy or not.

Creating ontology has many steps. Natalya F. Noy and Deborah L. McGuinnes from Standford University name these steps as below:

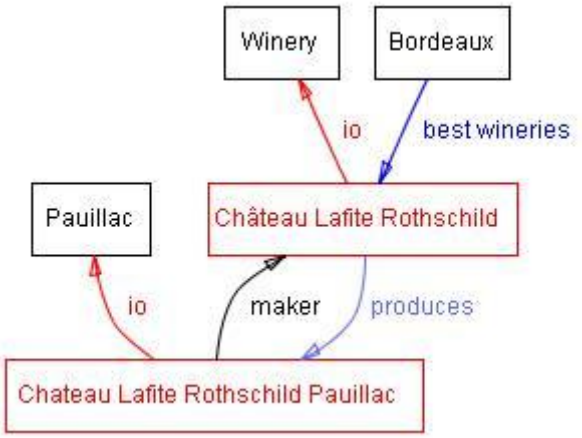
· defining classes in the ontology,

· arranging the classes in a taxonomic (subclass–superclass) hierarchy,

· defining slots and describing allowed values for these slots,

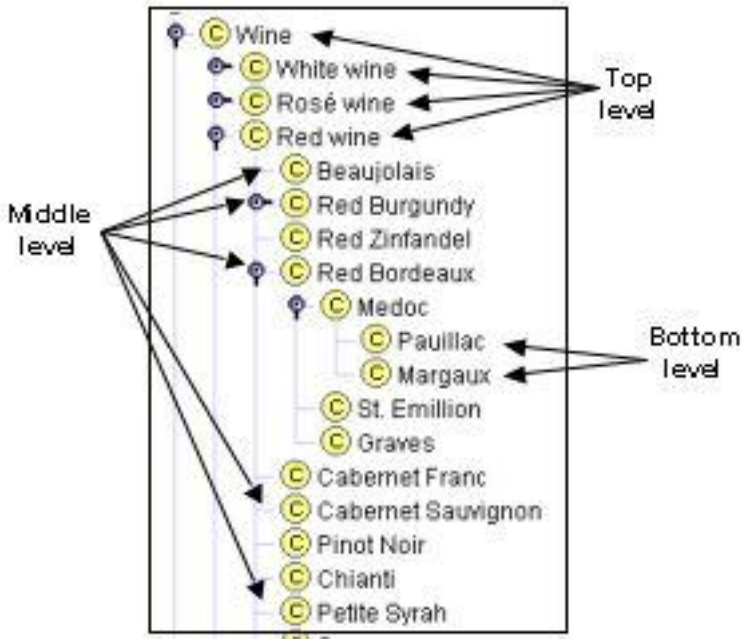
· filling in the values for slots for instances.

They gave wine example and figure in their assay. The figure below from their document (http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html) :



"Some classes, instances, and relations among them in the wine domain. We used black for classes and red for instances. Direct links represent slots and internal links such as instance-of and subclass-of."

Applying the steps above, we will create our own ontology for the project. We will have similar class hierarchy with the wine hierarchy:



4. SOFTWARE DEVELOPMENT MODEL

During decision period of the software development model which we will use for our project, we have learnt that in real life there is no company use the models exist strictly. In market, companies use the fusion of different methodologies. There is two main reasons for that. First, every single methodology has disadvantages. Therefore, company need to get rid of that disadvantage by using another model. Second, customer desires. Customers do not have to think about the model. They only want user friendly programs. Therefore, your model needs to be suitable for everyone.

Extreme Programming (XP) is a model which is popular nowadays. The most important property of that model is the customer is always with you during project. Every single step of the project customer can talk about the company wants. Therefore, you can make necessary adaptations during development period and at the end customer will get what he/she wants. However, that model has disadvantages too. All these disadvantages about the documentation because XP does not waste time with documentation at all. Therefore, your project design and development may not be reusable for other programmers. On the other hand, that model is for the programmers which are unfamiliar with project topic, and is for customers who do not know precisely they want. These two reasons help us to decide to use XP basically but not totally. Because we may need some fusion of models like companies in market.

http://www.stylusinc.com/Common/Concerns/SoftwareDevtPhilosophy.php

www.istanbul.edu.tr/Bolumler/enformatik/seminer/2004/seminer/XP.pps

5. PROJECT SCOPE

5.1.OBJECTIVE

The main objective of this project is text-mining on free-text radiology reports to get meaningful information, storing these in the database and integrating with the existing information system to make the patient records searchable.

5.2.DELIVERABLE

Project Proposal

Requirement Analysis Report

Initial Design Report

Final Design Report

5.3.MILESTONE

Milestone	**Date**
Tokenizer	30.11.2007
Morphologic Analyzer & POS	18.01.2008
Spell Checking	01.02.2008
Ontology	01.02.2008
Integration	08.02.2008
Machine Learning	25.04.2008
Training	23.05.2008

6. RESOURCES

We will use some open source and some licensed projects namely:

- **Text mining projects:**
  - Zemberek
  - Rapid Miner
  - Snomed
  - Protege
  - Jena

- **Database:**
  - Postgre SQL
  - PgAdmin
  - PgManager

- **Development Environments:**
  - NetBeans (GUI and as IDE)
  - MS Visual Studio 8.0

7. RISK ANALYSIS

7.1.RISK ASSESSMENT FORM

Risk Event	**Likelihood(0-6)**	**Impact(0-6)**
The time required to design, implement and test underestimated.	3	5
Not enough time to implement all the functions designed.	4	2
Not enough training data.	1	4

7.2.RISK RESPONSE MATRIX

Risk Event	**Contingency Plan**
The time required to design, implement and test underestimated.	Reschedule time plan
Not enough time to implement all the functions designed.	Set priorities to the functions and implement them with the priority order.
Not enough training data.	Try to find other sources.

8. MARKET ANALYSIS

From the researches we have made on the internet, we have found 2 examples similar to our project.

- Estard Data Miner 1.2.5

Estard Data Miner is a comprehensive data mining application, able to discover hidden relations both in structured and unstructured data.

CLASSIFICATION + IF-THEN RULES + DECISION TREES + STATISTICS + DECISION MODELS = ESTARD DATA MINER

Estard Data Miner Features

Estard Data Miner is based on unique **data mining algorithms.**
Some of the software functions are: importing data from various databases, statistical analysis, decision trees creation and revealing all rules which describe hidden correlations in data. The program allows to create reports on discovered knowledge.
You can create **decision rules**, revealing all the **if-then rules** in the data.
With the help of the program you can also build **decision trees.** Use them to process classification analysis.
The obtained decision rules and decision trees are **represented in a user-friendly, intuitive form**.
Statistical module contains charts and reports that are easy to understand, print and save.
Reports on decision rules, decision trees and statistical analysis are provided.
Rules can be edited or deleted in case if users want to combine their own knowledge with discovered one.
Wizards for data mining and data base loading will ease the process of data mining.
Different **analysis settings** for expert data mining customization are available.
Save the discovered rules, trees and statistics for further exploration and usage.
Use previously saved, uploaded or just obtained rules and decision trees to analyze databases, discovering classes within them.
3420 $

- Rapid miner

The modular operator concept of RapidMiner (formerly YALE) allows the design of complex nested operator chains for a huge number of learning problems in a very fast and efficient way (rapid prototyping). The data handling is transparent to the operators. They do not have to

cope with the actual data format or different data views - the RapidMiner core takes care of all necessary transformations. Read here about the most important features of RapidMiner.

The main **features** of RapidMiner are:

freely available **open-source** knowledge discovery environment

100% pure **Java** (runs on every major platform and operating system)

KD processes are modeled as simple **operator trees** which is both intuitive and powerful

operator trees or subtrees can be saved as **building blocks** for later re-use

internal **XML** representation ensures standardized interchange format of data mining experiments

simple scripting language allowing for automatic **large-scale** experiments

multi-layered data view concept ensures efficient and transparent data handling

Flexibility in using RapidMiner:

graphical user interface (GUI) for interactive prototyping

command line mode (batch mode) for automated large-scale applications

Java API (application programming interface) to ease usage of RapidMiner from your own programs

simple plugin and extension mechanisms, a broad variety of **plugins** already exists and you can easily add your own

powerful plotting facility offering a large set of sophisticated **high-dimensional visualization** techniques for data and models

more than 400 machine learning, evaluation, in- and output, pre- and post-processing, and visualization operators plus numerous meta optimization schemes

machine learning library **WEKA** fully integrated (WEKA web page)

RapidMiner was successfully applied on a wide range of applications where its rapid prototyping abilities demonstrated their usefulness, including **text mining**, **multimedia mining**, **feature engineering**, **data stream mining** and **tracking drifting concepts**, development of **ensemble methods**, and **distributed data mining**.

There is no buying price for commercial use but the course price for an individual is 1650 euros.

9. REQUIREMENTS

9.1. Software Requirements

- Java Virtual Machine
- SQL Server in the Server

9.2. Hardware Requirements

- 32 MB Graphics Card
- 512 MB Ram
- Pentium 4 or higher CPU

10. PROGRAMMING LANGUAGE

At the beginning of the Project we taught of two languages to write the program with , which were Java and C# because we had some experience of these languages. Then we decided to use Java for implementing our program with, since it is operating system independent and has a wide range of possible libraries which can be used in our program. We have decided to use NetBeans as our IDE since some of our group members had worked with that IDE before.

11. DATABASE DESIGN

In the design of the database, we have PATIENT, DOCTOR who have diagnosed the disease or illness of the PATIENT and the DISEASE and ABNORMAL which hold the names of the diseases and abnormalities. We have REPORT entity which holds a patientId and the doctorIds and the name of the report file. SENTENCE keeps all the sentences of the reports. SECTION keeps the information of the reports' sections like (Bulgular , Sonuç and etc. İn our datum). RESULT keeps the results of the report and if it is a disease or an abnormality.

- PATIENT
    - patientId (PK)
    - name
    - surname
    - age
    - gender

- DOCTOR
    - doctorId (PK)
    - name
    - surname
    - title
    - branch
- DISEASE
    - diseaseId (PK)
    - name
- ABNORMAL
    - abnormalId (PK)
    - existence
    - certainity
    - howDetermined
    - determinationDate
    - location
    - size
    - precision
    - dimension
    - sizeTrend
    - referenceDate
- SENTENCE
    - sentenceId (PK)
    - content
- SECTION
    - sectionId (PK)
    - name

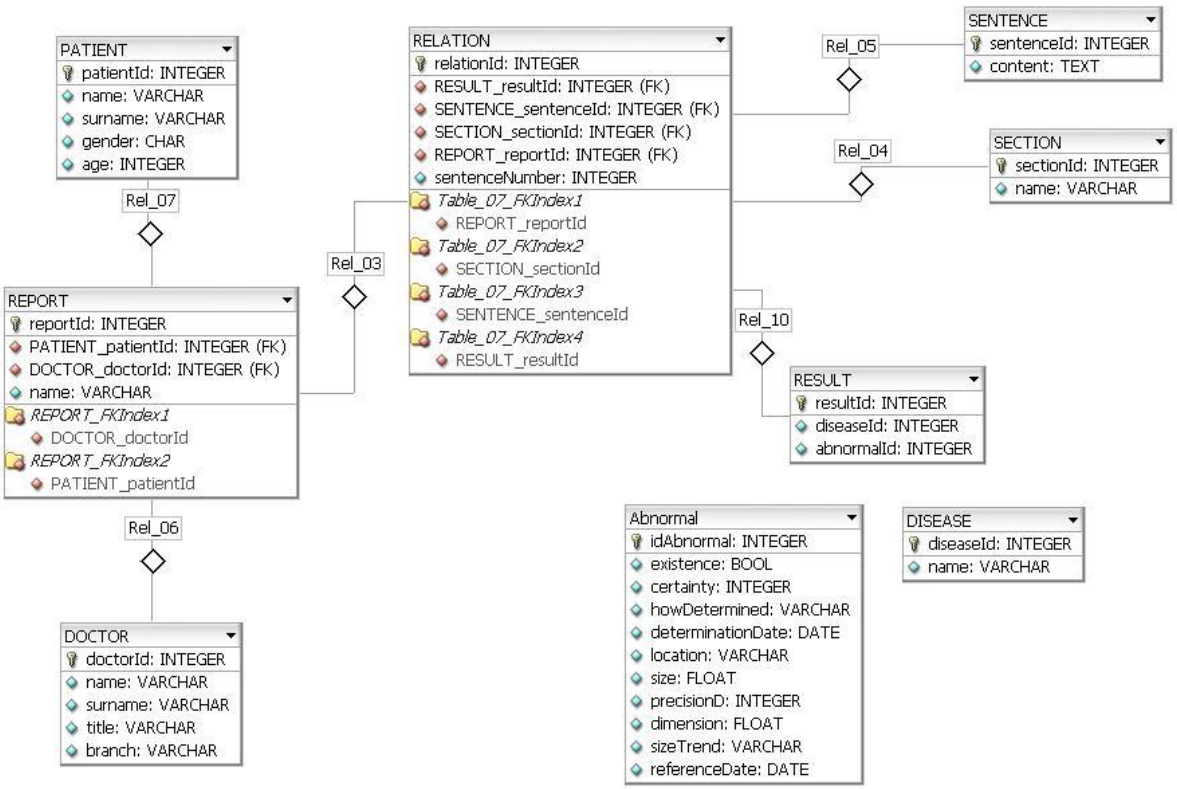
- RELATION
    - relationId (PK)
    - resultId
    - sentenceId
    - sectionId
    - reportId
    - sentenceNumber
- REPORT
    - reportId (PK)
    - patientId
    - doctorId
    - name
- RESULT
    - resultId (PK)
    - diseaseId
    - abnormalId

The ER diagram of the database is in Appendix.

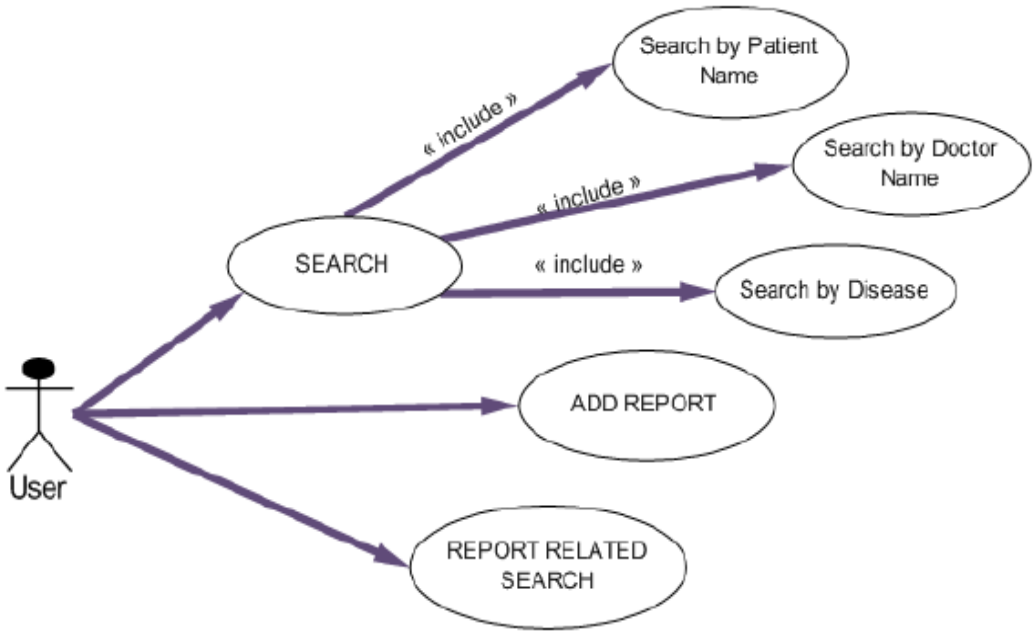
We can add some further attributes to entities.For example, some personal information for PATIENT entity.
So basically this is what our database will look like.

12. APPENDIX

12.1. ER Diagram

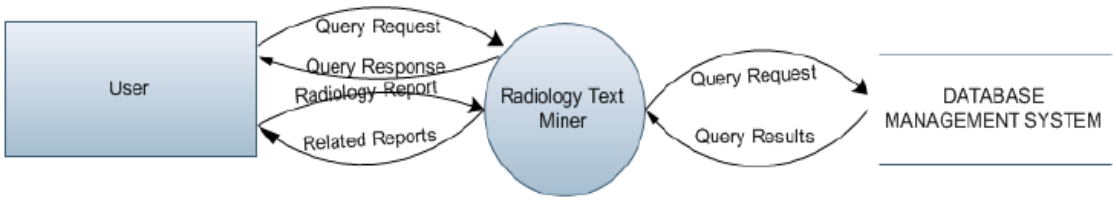


12.2. Use Case Diagram

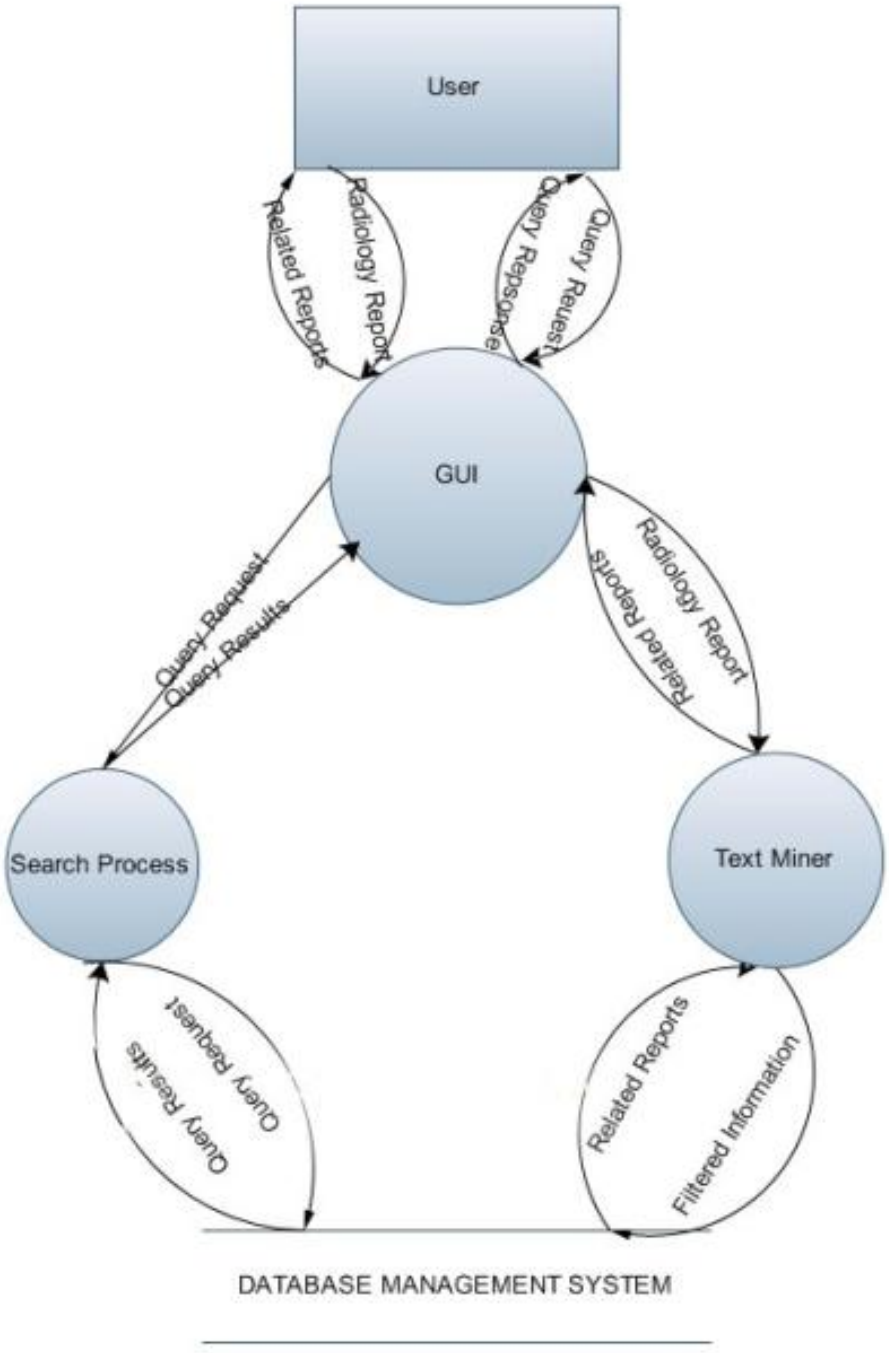


12.3. DataFlow Diagram

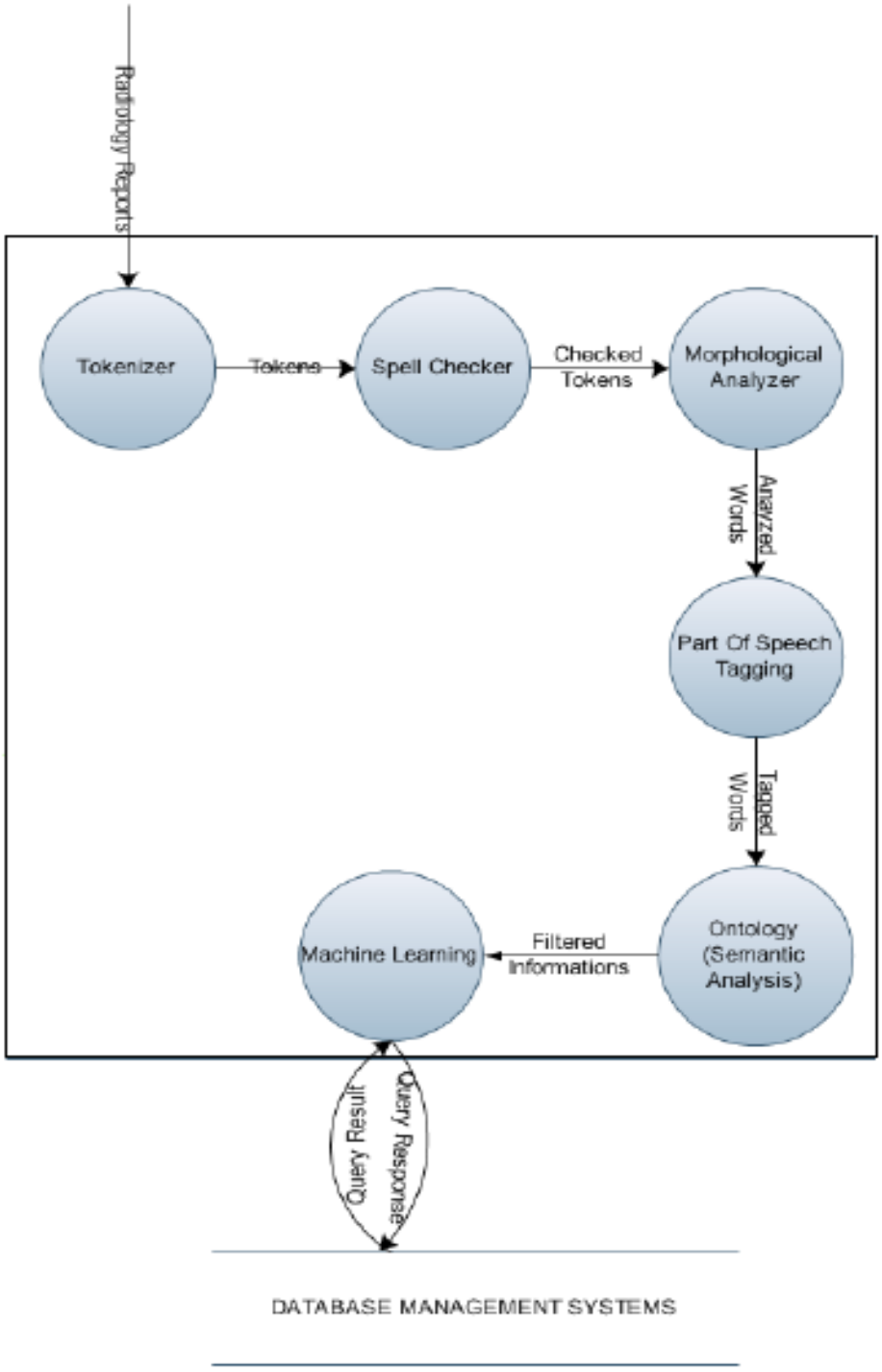
LEVEL 0



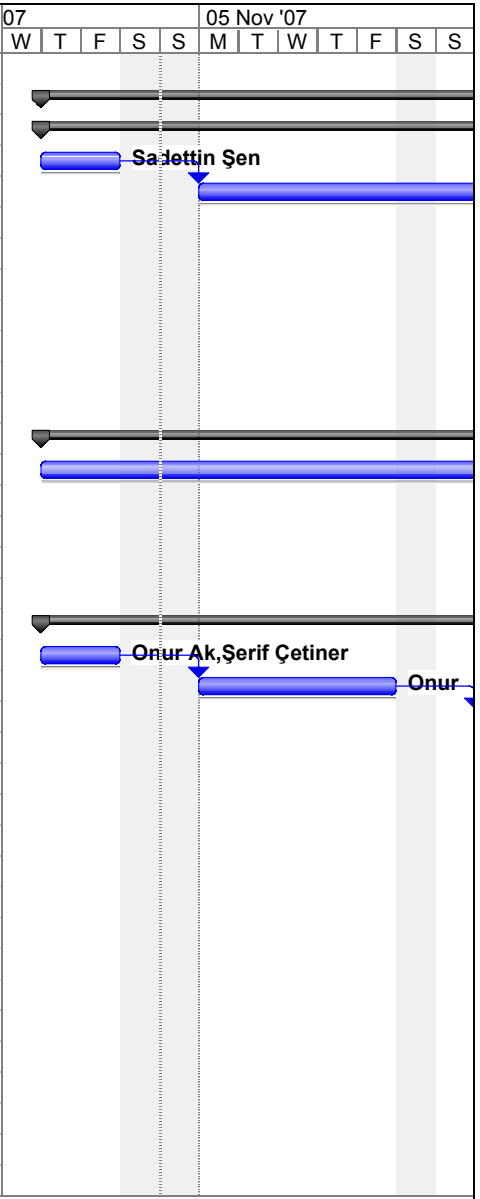
LEVEL 1


User
Related Reports
Radiology Report
Query Response
Query Request
GUI
Query Request
Query Results
Radiology Report
Related Reports
Search Process
Text Miner
Query Results
Query Request
Related Reports
Filtered Information
DATABASE MANAGEMENT SYSTEM

LEVEL2 TEXT MINER

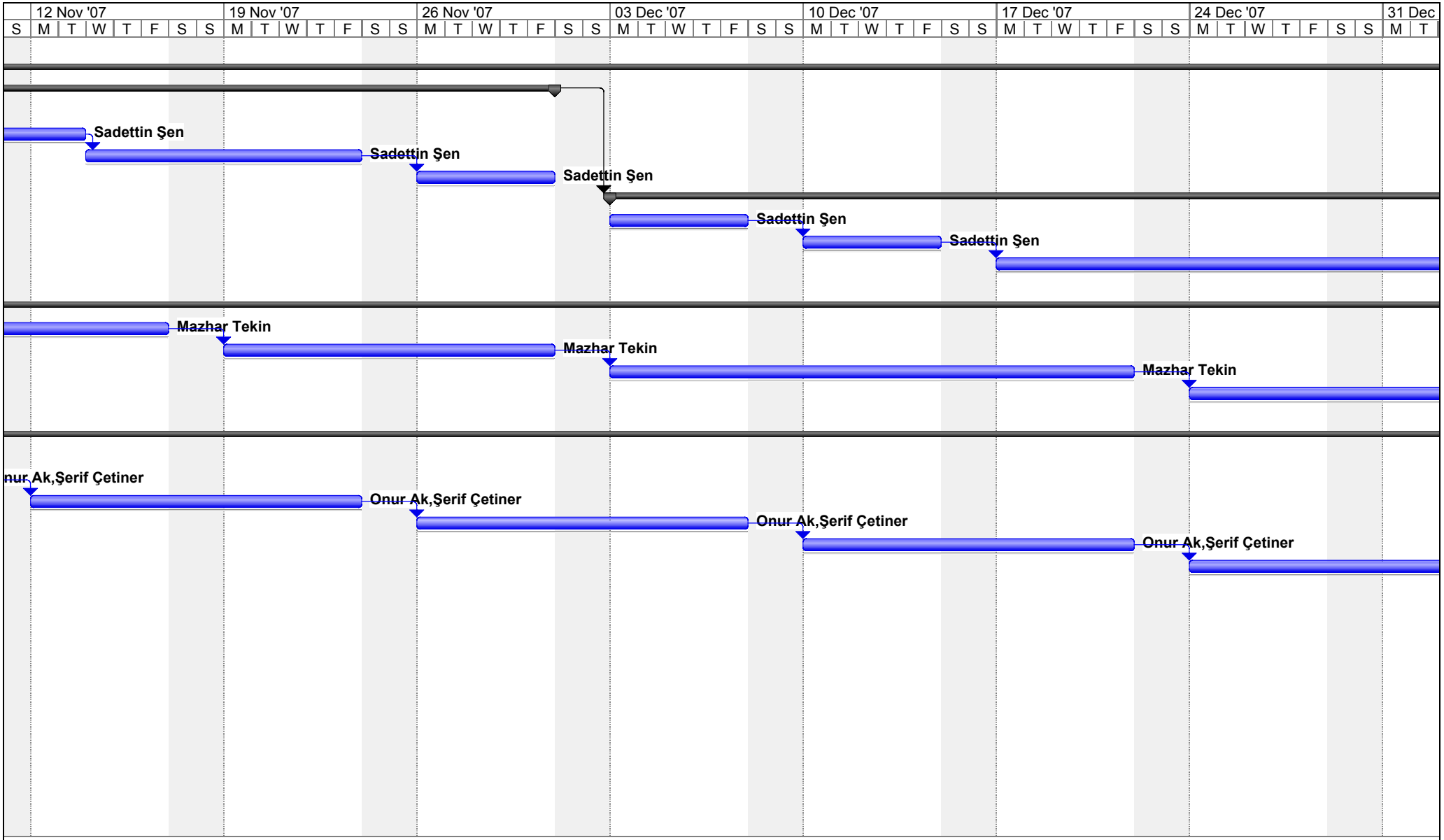


ID	Task Name	Duration	Start	Finish	Predecesso	Resource Names
1						
2	**Text Tokenizer & Spell Checking**	**67 days?**	**Thu 01.11.07**	**Fri 01.02.08**		**Sadettin Şen**
3	**Tokenizer**	**22 days?**	**Thu 01.11.07**	**Fri 30.11.07**		**Sadettin Şen**
4	Investigating Turkish ser	2 days?	Thu 01.11.07	Fri 02.11.07		Sadettin Şen
5	Finding a specific algorit	7 days?	Mon 05.11.07	Tue 13.11.07	4	Sadettin Şen
6	Writing the code for this	8 days?	Wed 14.11.07	Fri 23.11.07	5	Sadettin Şen
7	Testing & Debugging	5 days?	Mon 26.11.07	Fri 30.11.07	6	Sadettin Şen
8	**Spell Checking**	**45 days?**	**Mon 03.12.07**	**Fri 01.02.08**	**3**	**Sadettin Şen**
9	Investigating the spell-ch	5 days?	Mon 03.12.07	Fri 07.12.07		Sadettin Şen
10	Eliminating to several dif	5 days?	Mon 10.12.07	Fri 14.12.07	9	Sadettin Şen
11	Coding	20 days?	Mon 17.12.07	Fri 11.01.08	10	Sadettin Şen
12	Finding the best algorithı	15 days?	Mon 14.01.08	Fri 01.02.08	11	Sadettin Şen
13	**Morphological Analyzer & Part c**	**57 days?**	**Thu 01.11.07**	**Fri 18.01.08**		**Mazhar Tekin**
14	Searching "Zemberek" modul	12 days?	Thu 01.11.07	Fri 16.11.07		Mazhar Tekin
15	Constructing a medical corpu	10 days?	Mon 19.11.07	Fri 30.11.07	14	Mazhar Tekin
16	Syntactic analysis of medical	15 days?	Mon 03.12.07	Fri 21.12.07	15	Mazhar Tekin
17	Part of Speech Tagging	20 days?	Mon 24.12.07	Fri 18.01.08	16	Mazhar Tekin
18	**Database Design**	**10 days?**	**Mon 21.01.08**	**Fri 01.02.08**		**Mazhar Tekin**
19	**Ontological Process**	**67 days?**	**Thu 01.11.07**	**Fri 01.02.08**		**Onur Ak,Şerif Çetiner**
20	Determining the domain and s	2 days?	Thu 01.11.07	Fri 02.11.07		Onur Ak,Şerif Çetiner
21	Searching the availability of r	5 days?	Mon 05.11.07	Fri 09.11.07	20	Onur Ak,Şerif Çetiner
22	Enumerating important terms	10 days?	Mon 12.11.07	Fri 23.11.07	21	Onur Ak,Şerif Çetiner
23	Defining the classes and the	10 days?	Mon 26.11.07	Fri 07.12.07	22	Onur Ak,Şerif Çetiner
24	Defining the properties of clas	10 days?	Mon 10.12.07	Fri 21.12.07	23	Onur Ak,Şerif Çetiner
25	Defining the additional proper	10 days?	Mon 24.12.07	Fri 04.01.08	24	Onur Ak,Şerif Çetiner
26	Create instances	10 days?	Mon 07.01.08	Fri 18.01.08	25	Onur Ak,Şerif Çetiner
27	Create axioms/rules	10 days?	Mon 21.01.08	Fri 01.02.08	26	Onur Ak,Şerif Çetiner
28	**Integration**	**5 days?**	**Mon 04.02.08**	**Fri 08.02.08**	**2;13;18;19**	**Onur Ak,Şerif Çetiner,Sadetin Şen, Mazhar Tekin**
29	**Testing & Debugging**	**15 days?**	**Mon 11.02.08**	**Fri 29.02.08**	**28**	**Onur Ak,Şerif Çetiner,Sadetin Şen, Mazhar Tekin**
30	**Machine Learning**	**40 days?**	**Mon 03.03.08**	**Fri 25.04.08**	**29**	**Onur Ak,Şerif Çetiner,Sadetin Şen, Mazhar Tekin**
31	Learning Machine Learning A	10 days?	Mon 03.03.08	Fri 14.03.08		Onur Ak,Şerif Çetiner,Sadetin Şen, Mazhar Tekin
32	Determining the appropriate a	5 days?	Mon 17.03.08	Fri 21.03.08	31	Onur Ak,Şerif Çetiner,Sadetin Şen, Mazhar Tekin
33	Coding	20 days?	Mon 24.03.08	Fri 18.04.08	32	Onur Ak,Şerif Çetiner,Sadetin Şen, Mazhar Tekin
34	Testing	5 days?	Mon 21.04.08	Fri 25.04.08	33	Onur Ak,Şerif Çetiner,Sadetin Şen, Mazhar Tekin
35	**Testing & Debugging**	**10 days?**	**Mon 28.04.08**	**Fri 09.05.08**	**30**	**Onur Ak,Şerif Çetiner,Sadetin Şen, Mazhar Tekin**
36	**Training**	**10 days?**	**Mon 12.05.08**	**Fri 23.05.08**	**35**	**Onur Ak,Şerif Çetiner,Sadetin Şen, Mazhar Tekin**
37	**Web-Based User Interface**	**5 days?**	**Mon 26.05.08**	**Fri 30.05.08**		**Onur Ak,Şerif Çetiner**



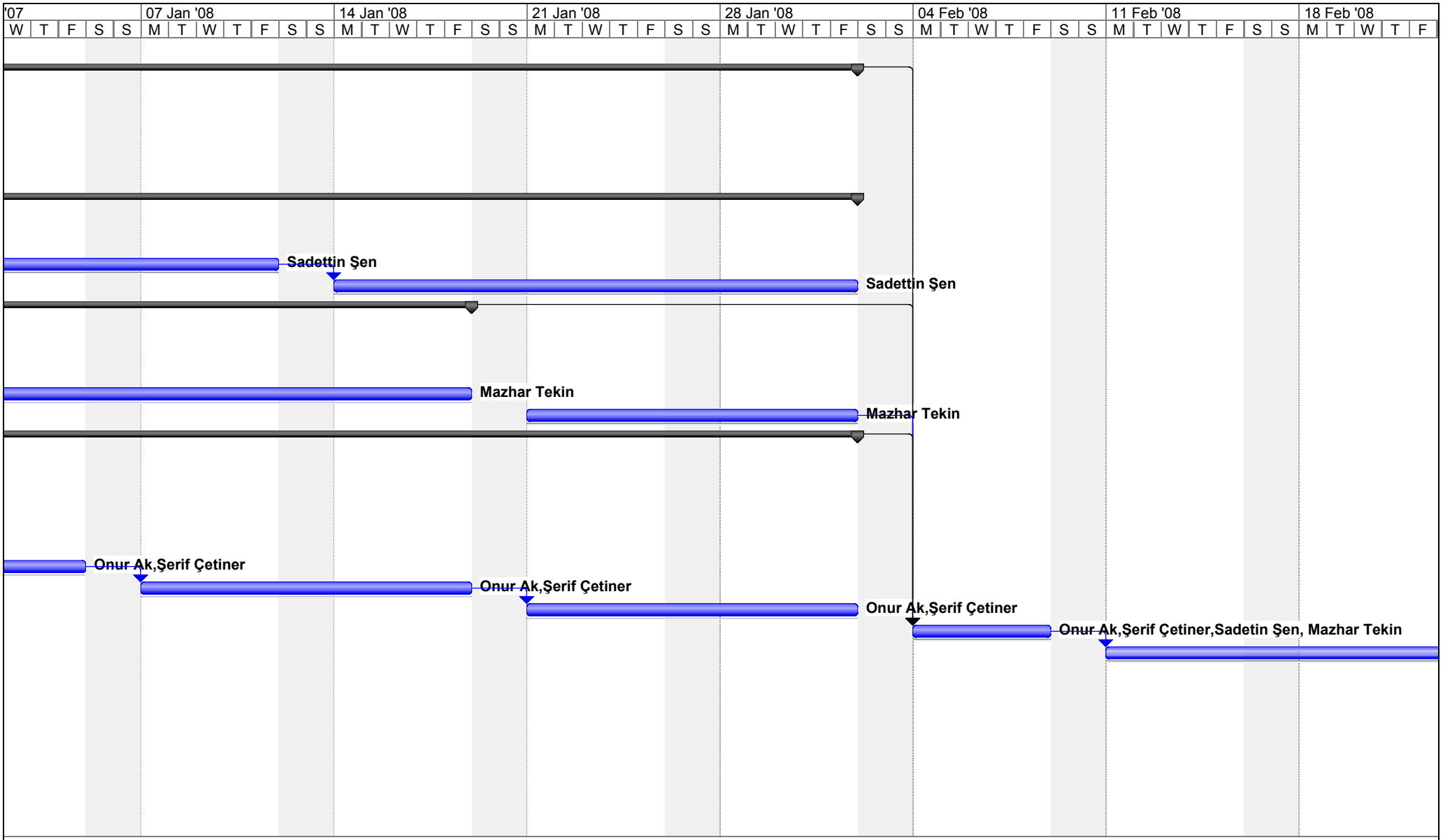
Project: Project1.mpp
Date: Sun 04.11.07

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline



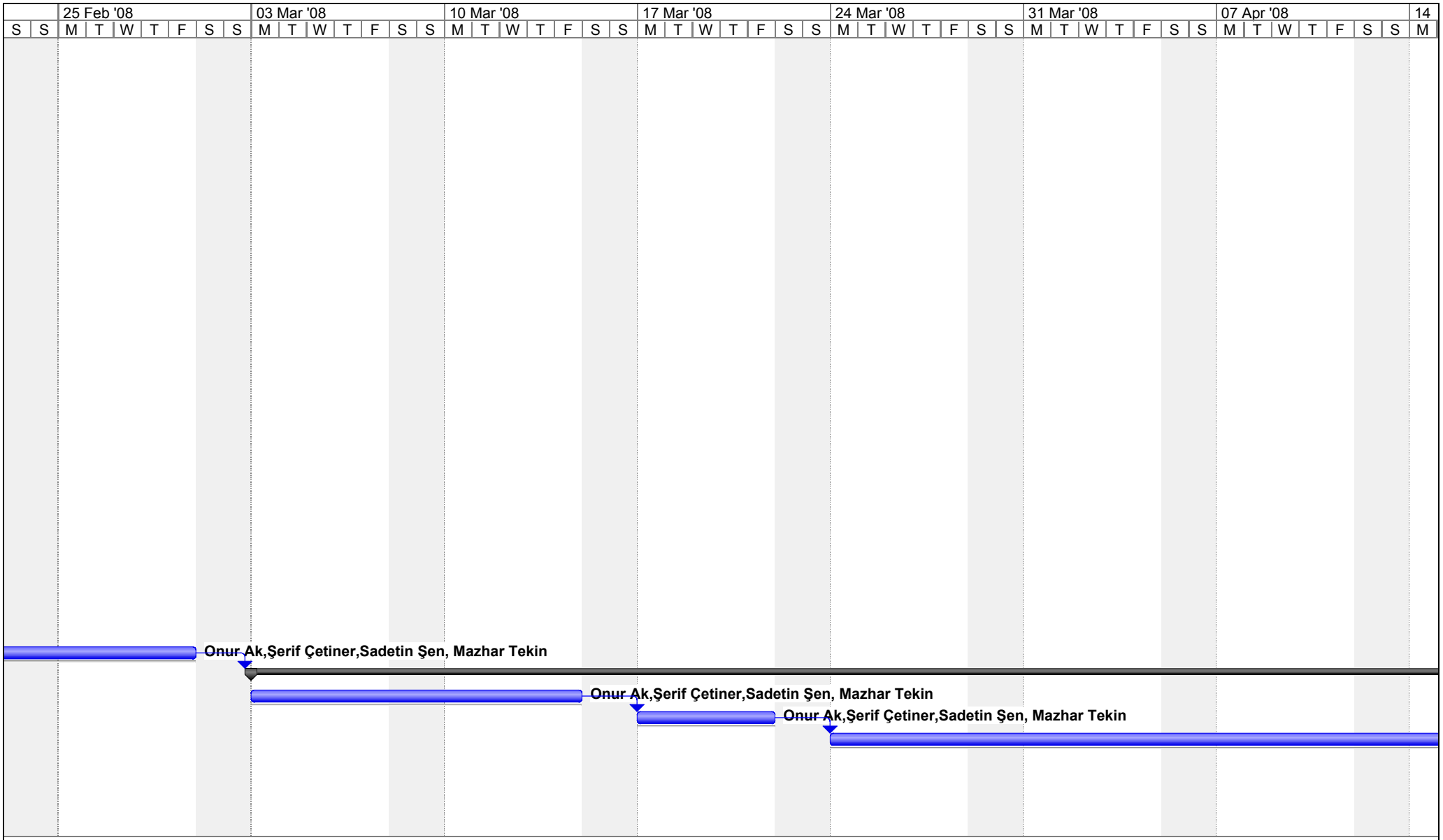
Project: Project1.mpp
Date: Sun 04.11.07

Task		Milestone		External Tasks	
Split		Summary		External Milestone	
Progress		Project Summary		Deadline	



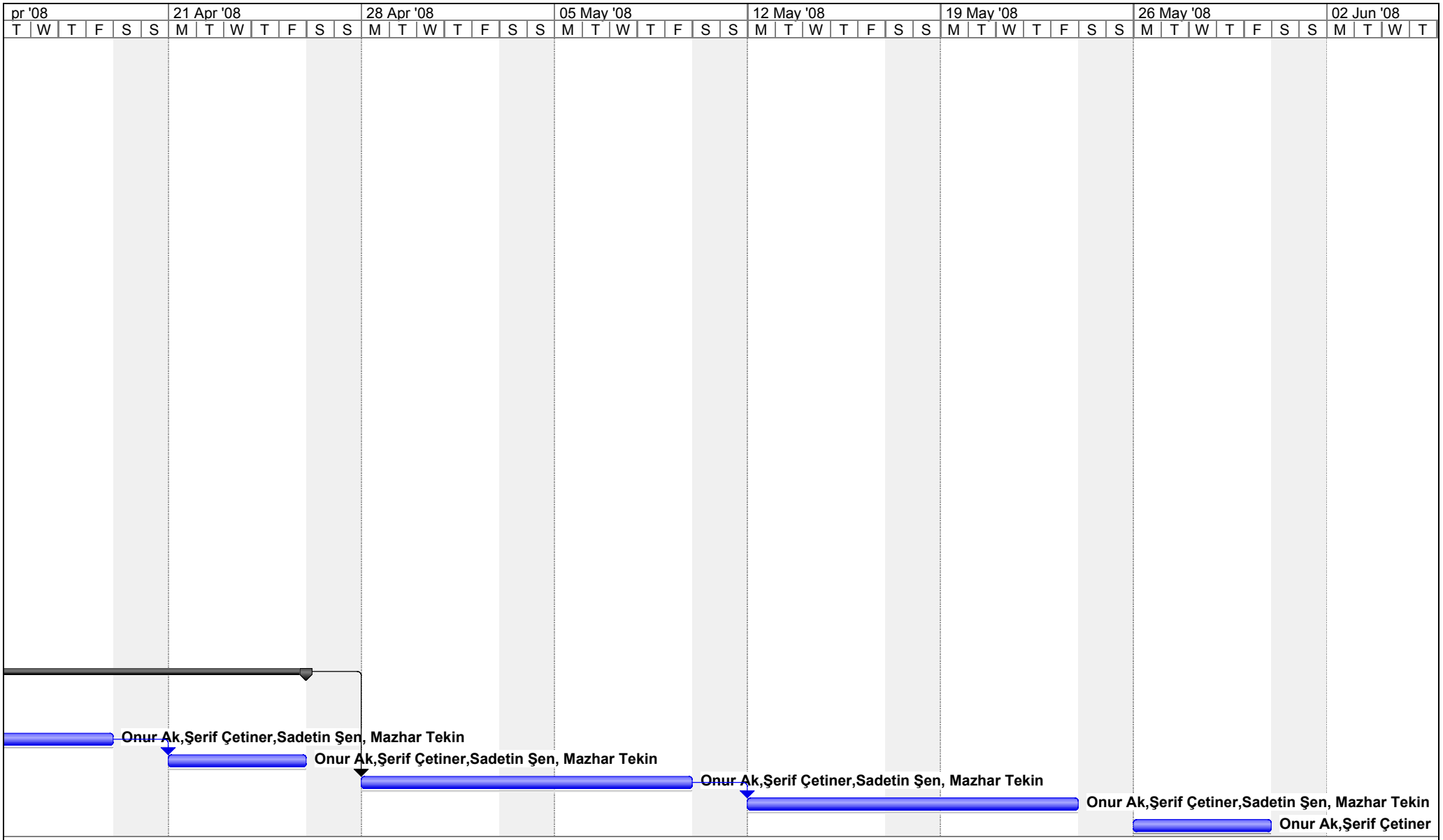
Project: Project1.mpp
Date: Sun 04.11.07

Task		Milestone		External Tasks	
Split		Summary		External Milestone	
Progress		Project Summary		Deadline	



Project: Project1.mpp
Date: Sun 04.11.07

Task		Milestone		External Tasks	
Split		Summary		External Milestone	
Progress		Project Summary		Deadline	



Project: Project1.mpp
Date: Sun 04.11.07

Task		Milestone		External Tasks	
Split		Summary		External Milestone	
Progress		Project Summary		Deadline	

13. REFERENCES

- http://www.linguistics.ruhr-uni-bochum.de/~strunk/ks2005FINAL.pdf
- http://ab.org.tr/ab06/bildiri/68.doc
- http://ab.org.tr/Yazismalar/2006/sunum/att-0251/01-E00-373161051.pdf
- http://www.icgst.com/AIML05/papers/P1120535133.pdf
- http://people.csail.mit.edu/jrennie/papers/ifile00.pdf
- http://en.wikipedia.org/wiki/Morphology_(linguistics)
- http://citeseer.ist.psu.edu/cache/papers/cs/129/ftp:zSzzSzftp.cs.bilkent.edu.trzSzpubzSztech-reportszSz1994zSzBU-CEIS-9423.pdf/using-a-corpus-for.pdf
- http://www.ida.liu.se/~janma/SemWeb/Slides/ontologies1.pdf
- http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html
- http://www.stylusinc.com/Common/Concerns/SoftwareDevtPhilosophy.php
- www.istanbul.edu.tr/Bolumler/enformatik/seminer/2004/seminer/XP.pps