AUTOMATING INFORMATION EXTRACTION TASK FOR TURKISH TEXTS

A DISSERTATION SUBMITTED TO
THE DEPARTMENT OF COMPUTER ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Serhan Tatar
January, 2011

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. Dr. Özgür Ulusoy (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Dr. İlyas Çiçekli (Co-Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. Dr. Fazlı Can

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Assoc. Prof. Dr. Ferda Nur Alpaslan

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Asst. Prof. Dr. Selim Aksoy

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Asst. Prof. Dr. İbrahim Körpeoğlu

Approved for the Institute of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Institute

ABSTRACT

AUTOMATING INFORMATION EXTRACTION TASK FOR TURKISH TEXTS

Serhan Tatar
Ph.d. in Computer Engineering
Supervisors: Prof. Dr. Özgür Ulusoy and Dr. İlyas Çiçekli
January, 2011

Throughout history, mankind has often suffered from a lack of necessary resources. In today's information world, the challenge can sometimes be a wealth of resources. That is to say, an excessive amount of information implies the need to find and extract necessary information. Information extraction can be defined as the identification of selected types of entities, relations, facts or events in a set of unstructured text documents in a natural language.

The goal of our research is to build a system that automatically locates and extracts information from Turkish unstructured texts. Our study focuses on two basic Information Extraction (IE) tasks: Named Entity Recognition and Entity Relation Detection. Named Entity Recognition, finding named entities (persons, locations, organizations, etc.) located in unstructured texts, is one of the most fundamental IE tasks. Entity Relation Detection task tries to identify relationships between entities mentioned in text documents.

Using supervised learning strategy, the developed systems start with a set of examples collected from a training dataset and generate the extraction rules from the given examples by using a carefully designed coverage algorithm. Moreover, several rule filtering and rule refinement techniques are utilized to maximize generalization and accuracy at the same time. In order to obtain accurate generalization, we use several syntactic and semantic features of the text, including: orthographical, contextual, lexical and morphological features. In particular, morphological features of the text are effectively used in this study to increase the extraction performance for Turkish, an agglutinative language. Since the system does not rely on handcrafted rules/patterns, it does not heavily suffer from domain adaptability problem.

The results of the conducted experiments show that (1) the developed systems

are successfully applicable to the Named Entity Recognition and Entity Relation Detection tasks, and (2) exploiting morphological features can significantly improve the performance of information extraction from Turkish, an agglutinative language.

Keywords: Information Extraction, Turkish, Named Entity Recognition, Entity Relation Detection.

ÖZET

TÜRKÇE METİNLERDEN OTOMATİK BİLGİ ÇIKARIMI

Serhan Tatar
Bilgisayar Mühendisliği, Doktora
Tez Yöneticileri: Prof. Dr. Özgür Ulusoy ve Dr. İlyas Çiçekli
Ocak, 2011

Tarih boyunca, kaynakların yetersizliği insanoğlu için sorun olmuştur. Ne var ki günümüz bilgi dünyasında, kaynakların yetersizliğinden ziyade kaynak fazlalığının sebep olduğu yeni bir problem türüyle karşı karşıyayız. Aşırı bilgi, ihtiyaç duyulan bilginin bulunmasını ve çıkarımını gerektirmektedir. Bilgi çıkarımı, ihtiyaç duyulan nesnelerin, ilişkilerin, gerçeklerin veya olayların, doğal dildeki serbest metinler içerisinde bulunması olarak tanımlanabilir. Bu bağlamda bilgi çıkarımı, doğal dildeki yapısal olmayan metinlerin çözümlenmesi ve bu metinlerin ihtiva ettiği gerekli bilginin yapısal bir şablona aktarılması işlemidir.

Bu çalışmanın amacı Türkçe serbest metinlerdeki bilgiyi otomatik olarak bulan ve çıkaran bir sistemin geliştirilmesidir. Çalışma iki temel bilgi çıkarımı görevine odaklanmaktadır: Ad Tanıma ve İlişki Bulma. En temel bilgi çıkarımı görevlerinden olan Ad Tanıma, serbest metinlerde geçen varlık isimlerinin (insan, yer, organizasyon vb.) bulunmasıdır. İlişki Bulma görevi ise, metinlerde bahsedilen varlıklar arasındaki ilişkileri bulmaya çalışır.

Gözetimli ögrenme stratejisini kullanan sistem, öğrenme kümesinden seçilen örnek kümesi ile başlayıp bilgi çıkarım kurallarını üretmektedir. Ayrıca, genelleştirmenin ve doğruluğun maksimize edilmesi amacıyla kural filtreleme ve kural iyileştirme teknikleri kullanılmaktadır. Hassas genelleştirmenin sağlanması maksadıyla imla, bağlam, sözcük, biçim gibi çeşitli sözdizimsel ve anlamsal metin özelliklerinden faydalanılmaktadır. Özellikle, bitişimli bir dil olan Türkçe'den bilgi çıkarımı başarımının artırılması için biçimbilimsel özellikler etkin olarak kullanılmıştır. Sistem elle üretilen kurallar üzerine dayanmadığı için alan uyumluluğu probleminden ciddi olarak etkilenmemektedir.

Yapılan test sonuçları, (1) geliştirilen sistemin Ad Tanıma ve İlişki Bulma

görevlerine başarılı bir şekilde uygulandığını, ve (2) biçimbilimsel özelliklerin kullanımının, bitişimli bir dil olan Türkçe'den bilgi çıkarımı işleminin performansını önemli ölçüde artırdığını göstermiştir.

Anahtar sözcükler: Bilgi Çıkarımı, Türkçe, Ad Tanıma, İlişki Bulma.

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my advisor Dr. İlyas Çiçekli for his guidance, patience, and active support during this long journey. I would also like to thank Prof. Dr. Özgür Ulusoy for his support.

I would like to thank the members of my thesis committee Prof. Dr. Fazlı Can, Assoc. Prof. Dr. Ferda Nur Alpaslan, Asst. Prof. Dr. Selim Aksoy, and Asst. Prof. Dr. İbrahim Körpeoğlu for their invaluable comments.

I consider myself fortunate to have had the chance to take courses from distinguished faculty members throughout my doctoral study. I am grateful to Asst. Prof. Dr. Selim Aksoy, Prof. Dr. Cevdet Aykanat, Prof. Dr. H.Altay Güvenir, Asst. Prof. Dr. İbrahim Körpeoğlu, Prof. Dr. Bülent Özgüç, and Asst. Prof. Dr. Ali Aydın Selçuk. I am also indebted to them for their excellent research and teaching, which have significantly influenced me.

I would like to thank the scientists at the Defence Research and Development Canada - Atlantic Center for their help and support during my research visit (Canadian Defence Research Fellowship Program) between September 2006 and September 2007. I want to express my special thanks to David Chapman. It was a pleasure to work with such professional people.

I would like to thank Mücahid Kutlu for Turkish Morphological Disambiguator used in this study.

Doctoral study is a challenging task. Having professional commitments and responsibilities in my military career has made mine even more challenging. I am grateful to LtCol. Ramazan Ercan, LtCol. Cemal Gemci, Col. Bülend Ayyıldız, Col. Şükrü Kısadere, Col. Fikret Serbest, Col. Bilgehan Doruk, CDR. Andrew Mason, and Mr. Pall Arnason for their support.

I thank my friends Şahin Yeşil, Ümit Altıntakan, Mahmut Bilgen, Ziya Bayrak, Ata Türk, Rıfat Özcan, Aydemir Memişoğlu, Hüseyin Özgür Tan, and Ozan Alptekin for their friendship and support.

I would like to thank my brother Erhan Tatar and my cousin Ünal Tatar for their brotherhood.

Lastly, I would like to thank my parents for believing in me and for encouraging me throughout my life. Without their support, this thesis would not have been possible.

Serhan TATAR

Anneme ve Babama.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Information Extraction

Recently, we have observed an explosive growth in the amount of available information. Especially with the advances in computer technology and the popularization of the Internet, there has been an exponential increase in the number of online resources. As estimated in [64], 1.5 exabytes (1.5 billion gigabytes) of storable information was produced in 1999. According to the report, this is equivalent to about 250 megabytes for every man, woman, and child on earth. Thus, the vast amount of information is accessible to an ordinary person today. For most of the people, idea of having more available resources than the needed amount may seem preferable. However, it is not easy for an individual to search all documents in order to find the specific piece of information that she/he needs. Therefore, excessive amount of information brings a new type of problem into existence: finding and extracting necessary information.

As in many cases, computer assistance can be used to overcome the problem. Information retrieval (IR) aims to develop automatic methods for indexing large document collections and searching for documents in those collections, for the information within the documents. Current research in information retrieval makes it possible to retrieve relevant documents from a document collection.

However most of the information is in human languages, not in databases or other structured formats, and unfortunately, interpreting natural language texts is a task that humans are simply better suited for than computers.

Natural language processing (NLP), a sub-field of artificial intelligence and linguistics, focuses on the automated systems that can analyze, understand, and generate natural human languages. It addresses many tasks to understand the meaning of the speech/text in natural languages and translate them into machine understandable representations. The ultimate goal is to manipulate the information in more user-friendly ways (e.g. controlling aircraft systems by voice commands) by using the computational power of machines.

Among the others, information extraction (IE) is an important task in the field. IE has the main goal of automating the process of finding valuable pieces of information out of huge data. We should distinguish IE from a number of major research fields. IR retrieves relevant documents from a document collection, whereas IE retrieves relevant information from documents. Question answering (QA), in which the system first finds relavant documents and then extracts the asked information from the retrieved documents, can be seen as the combination of IR and IE. Both IE and data mining (DM) search for the information available in the documents. However, DM aims to discover or derive new information from data [44], while IE focuses on the extraction of the information already available in the documents.

1.1.1 What is IE?

Basically, information extraction can be defined as the identification of selected types of entities, relations, facts or events in a set of unstructured text documents in a natural language. It is the process of analyzing unstructured texts and extracting the necessary information into a structured representation, or as described in [38] - the process of selective information structuring. IE transforms free text into a structured form and reduces the information in a document to a tabular structure and does not attempt to understand whole document.

As stated in the previous section, information extraction is an important task in the NLP field. However, IE owns some features that make the task more manageable when compared to many other NLP tasks. First of all, the task does not care about author's intentions and need to answer general questions about documents. The aim is to populate the slots of the defined template. Therefore, a less expressive representation of the meaning of a document can be sufficient for IE. Moreover, IE is a well-defined task; we know what we search for and how we encode the output information.

Before giving a formal definition of the problem, it is helpful to give a few examples. A simple example may be automatic discovery and extraction of protein names from biological texts. An example text from YAPEX [28] corpora, whose protein names are marked, is shown in Figure- 1.1. In the corpora each article has four sections:

- MedlineID: starts with <MedlineID> tag and ends with </MedlineID>.
- PMID: starts with <PMID> tag and ends with </PMID>.
- ArticleTitle: starts with <ArticleTitle> tag and ends with </ArticleTitle>.
- AbstractText: starts with <AbstractText> tag and ends with </AbstractText>.

Last two parts, ArticleTitle and AbstractText, contain protein names. In the figure, tagged protein names can be seen clearly. Each protein name is marked by two tags: <Protname> and </Protname> (e.g. <Protname> retinoic acid receptor alpha </Protname>). In the example, target entities are proteins. A simple extractor may learn rules from the tagged biological texts and extract protein names from un-tagged texts by using the generated rules.

A more complex example may describe the levels of detail that systems can extract. Figure- 1.2 shows a sample input text where the necessary information lies. In the example, a news article [21] is presented. We can extract different kind of information from the story. For instance, the entities (an object of interest

<**PubmedArticle**>

<**MedlineID**>21294781</**MedlineID**>

<**PMID**>11401507</**PMID**>

<**ArticleTitle**>Molecular dissection of the <Protname>importin beta1</Protname>-recognized nuclear targeting signal of <Protname>parathyroid hormone-related protein</Protname>.</**ArticleTitle**>

<**AbstractText**>Produced by various types of solid tumors, <Protname>parathyroid hormone-related protein</Protname> (<Protname>PTHrP</Protname>) is the causative agent of humoral hypercalcemia of malignancy. The similarity of <Protname>PTHrP's</Protname> amino-terminus to that of <Protname>parathyroid hormone</Protname> enables it to share some of the latter's signalling properties, but its carboxy-terminus confers distinct functions including a role in the nucleus/nucleolus in reducing apoptosis and enhancing cell proliferation. <Protname>PTHrP</Protname> nuclear import occurs via a novel <Protname>importin beta1</Protname>-mediated pathway. The present study uses several different direct binding assays to map the interaction of <Protname>PTHrP</Protname> with <Protname>importin beta</Protname> using a series of alanine mutated <Protname>PTHrP</Protname> peptides and truncated human <Protname>importin beta1</Protname> derivatives. Our results indicate that <Protname>PTHrP</Protname> amino acids 83-93 (KTPGKKKKGK) are absolutely essential for <Protname>importin beta1</Protname> recognition with residues 71-82 (TNKVETYKEQPL) additionally required for high affinity binding; residues 380-643 of <Protname>importin beta1</Protname> are required for the interaction. Binding of <Protname>importin beta1</Protname> to <Protname>PTHrP</Protname> is reduced in the presence of the GTP-bound but not GDP-bound form of the guanine nucleotide binding protein <Protname>Ran</Protname>, consistent with the idea that <Protname>Ran</Protname>GTP binding to <Protname>importin beta</Protname> is involved in the release of <Protname>PTHrP</Protname> into the nucleus following translocation across the nuclear envelope. This study represents the first detailed examination of a modular, non-arginine-rich <Protname>importin beta1</Protname>-recognized nuclear targeting signal. Copyright 2001 Academic Press.</**AbstractText**>

</**PubmedArticle**>

Figure 1.1: Sample Tagged Medline Abstract

Fletcher Maddox, former Dean of the UCSD Business School, announced the formation of La Jolla Genomatics together with his two sons. La Jolla Genomatics will release its product Geninfo in June 1999. Geninfo is a turnkey system to assist biotechnology researchers in keeping up with the voluminous literature in all aspects of their field.

Dr. Maddox will be the firm's CEO. His son, Oliver, is the Chief Scientist and holds patents on many of the algorithms used in Geninfo. Oliver's brother, Ambrose, follows more in his father's footsteps and will be the CFO of L.J.G. headquartered in the Maddox family's hometown of La Jolla, CA.

Figure 1.2: Sample News Article

such as a person or organization) and attributes associated with them extracted from the text are shown below.

- ***ENTITY*** **{** NAME = "Fletcher Maddox" ; DESCRIPTOR = "Former Dean of USCD Business School" ; TYPE = Person; **}**
- ***ENTITY*** **{** NAME = "Dr. Maddox"; DESCRIPTOR = "his father "; DESCRIPTOR = " the firm's CEO "; TYPE = Person; **}**
- ***ENTITY*** **{** NAME = "Oliver"; DESCRIPTOR = "His son"; DESCRIPTOR = "Chief Scientist"; TYPE = Person; **}**
- ***ENTITY*** **{** NAME = "Ambrose"; DESCRIPTOR = "Oliver's brother"; DESCRIPTOR = "the CFO of L.J.G."; TYPE = Person; **}**
- ***ENTITY*** **{** NAME = "UCSD Business School"; TYPE = Organization; **}**
- ***ENTITY*** **{** NAME = "La Jolla Genomatics"; TYPE = Organization; **}**
- ***ENTITY*** **{** NAME = "L.J.G."; TYPE = Organization; **}**
- ***ENTITY*** **{** NAME = "Geninfo"; DESCRIPTOR = "its product"; TYPE = Artifact; **}**
- ***ENTITY*** **{** NAME = "La Jolla"; DESCRIPTOR = "the Maddox family's hometown"; TYPE = Location; **}**

- ***ENTITY*** **{** NAME = "CA"; TYPE = Location; **}**
- ***ENTITY*** **{** NAME = "June 1999"; TYPE = Date; **}**

Relations between the extracted entities (or facts) can be the target of information extraction.

- ***RELATION*** **{** ENTITY_1 = "Fletcher Maddox"; ENTITY_2 = "UCSD Business School"; TYPE = Employee_of; **}**
- ***RELATION*** **{** ENTITY_1 = "Fletcher Maddox"; ENTITY_2 = "La Jolla Genomatics"; TYPE = Employee_of; **}**
- ***RELATION*** **{** ENTITY_1 = "Oliver"; ENTITY_2 = "La Jolla Genomatics"; TYPE = Employee_of; **}**
- ***RELATION*** **{** ENTITY_1 = "Ambrose"; ENTITY_2 = "La Jolla Genomatics"; TYPE = Employee_of; **}**
- ***RELATION*** **{** ENTITY_1 = "Geninfo"; ENTITY_2 = "La Jolla Genomatics"; TYPE = Product_of; **}**
- ***RELATION*** **{** ENTITY_1 = "La Jolla"; ENTITY_2 = "La Jolla Genomatics"; TYPE = Location_of; **}**
- ***RELATION*** **{** ENTITY_1 = "CA"; ENTITY_2 = "La Jolla Genomatics"; TYPE = Location_of; **}**
- ***RELATION*** **{** ENTITY_1 = "La Jolla"; ENTITY_2 = "CA"; TYPE = Location_of; **}**

We can also extract the events available in the text. Events extracted from the example text are shown below.

- ***EVENT*** **{** PRINCIPAL = "Fletcher Maddox"; DATE = " "; CAPITAL = ""; TYPE = Company_Formation; **}**

- ***EVENT*** **{** COMPANY = "La Jolla Genomatics"; PRODUCTS = "Geninfo"; DATE = "June 1999"; COST = " "; TYPE = Product_Release; **}**

1.1.2 Formal Definition

After examining several examples of information extraction, we can give a formal definition of the problem. We will follow the machine learning approach described in [34]. Information extraction task takes two inputs: a knowledge source and a predefined template. The output of the task is semantically explicit information suitable for the given template.

The first input, knowledge source, is a collection of documents. Let D represent a document in the input collection. D can be seen as a sequence of terms, $\langle t_1, \cdots, t_n \rangle$, where a term is an atomic processing unit (e.g. a word, a number, or a unit of punctuation).

The second input, target template, can be seen as a collection of fields where a field is a function, $F(D) = \{(i_1, j_1), (i_2, j_2), \cdots, (i_n, j_n)\}$, mapping a document to a set of fragments from the document. In the definition, i_k and j_k are the location index values of the left and right boundaries of fragment k ($k \leq n$). If input document does not include a specific field, $F(D)$ returns the empty set.

One way of looking to problem is to find a function F' that approximates F as well as possible and generalizes to unseen documents. An alternative way to this approach is using a function, $G(D, i, j)$ which maps a document sub-sequence to a real number representing the system's confidence whether a text fragment (i, j) is a field instance. In this way, the problem is reduced to task of presenting G with fragments of appropriate size, and picking the fragment for which G's output is highest. Moreover, we also want to use G to reject some fragments. This can be accomplished by associating a threshold with G.

1.1.3 Common IE Tasks

IE is a multilateral research area. The tasks performed by IE systems usually differ, but the following are some of the common IE tasks:

- Named Entity Recognition (NER) task deals with locating the entities (persons, locations, organizations, etc.) in the text.
- Entity Relation Detection (ERD) task requires identifying relationships between entities (e.g. PRODUCT_OF, EMPLOYEE_OF).
- Event Extraction (EE) task requires identifying instances of a task-specific event in which entities participate and identifying event-attributes.

1.1.4 Language Impact

The characteristics of the source language to extract information from also have a significant impact on the extraction techniques being used. A certain feature of one language, which can help the extraction process, may not be available for another one. For example, unlike English, there are no spaces between words in Chinese, which makes a text segmentation process essential prior to IE [104]. Chinese and Arabic further lack the capitalization information which can be used as clues for identifying named entities [104, 10]. Absence of short vowels is yet another difficulty in IE from Arabic texts since it renders the lexical items a lot more ambiguous than in other languages aggravating the homography problem [10]. Moreover, a language specific phenomenon can complicate the IE task. For instance, in German, all nouns are capitalized; consequently the number of word forms to be considered as potential named entities is much larger [83]. In Slavonic languages the case of the noun phrase within a numerical phrase depends on the numeral and on the position of the whole numerical phrase in the sentence [79]. Likewise, IE for the languages with complex morphological structures, such as Turkish, requires a morphological level of processing.

1.1.5 Domain Adaptability/Portability

One of the key challenges in the IE field is domain adaptability/portability. Domain adaptation can be described as the process of adapting an extraction system developed for one domain to another domain. As for the domain itself, it can be thought of as the genre and format of the content in documents from which named entities will be extracted. To illustrate: how easy can a system developed for extracting people names from news articles be adapted for extracting people names from seminar announcements? Can a system designed for the identification of person names locate protein names in biomedical text? In fact, adapting a system to a new domain can sometimes be compared to developing a new system altogether. That is to say, adapting knowledge-source based and rule-based IE approaches to new domains is generally not straightforward since it essentially requires human intervention to first analyze the domain and develop the appropriate resources to tackle it (i.e. dictionaries, rules etc.). Furthermore, keeping these resources up-to-date given evolution in domains also requires constant human intervention.

1.1.6 Application Areas

Possible application areas of the IE research include a variety of fields. Security and intelligence is an important application area where the rich interpretation provided by IE is needed. To perform intelligence research and analysis effectively, IE can be used in an efficient manner. In intelligence analysis, entities are key pieces of information, such as people, places, phone numbers and addresses. Information extraction helps analysts and field personnel automatically identify, extract and classify mission-critical entities, relations between or among entities, and the multiple aspects of events from unstructured text to provide faster, more accurate intelligence. Thus, information extraction is an essential tool for operations that require link analysis, event tracking and order of battle analysis.

Another application field of the research may be business world. Competitive intelligence is an important organizational function responsible for the early

identification of risks and opportunities in the market. To know what others know and what others do provide great advantage in the competitive environment of business world. Current IE technology can be used in competitive intelligence by enabling actors in the business world to monitor their competitors' activities on open information sources. The capability of processesing large volumes of data, recognizing, interpreting, and extracting entities, relations, and events of interest can serve analysts, executives and managers in decision making process.

Biomedical domain is just another application area for IE methods. Biological knowledge, generated as a result of biological research, is currently stored in scientific publications which can be accessed via different knowledge sources storing vast amounts of information - Medline[1] being a prominent example. Knowledge sources do not, however, feature a formal structure in which to access stored information, thus rendering information search, retrieval and processing especially tedious and time-consuming. This consequently results in a strong demand for automatized discovery and extraction of information.

IE can also be beneficial in the currently developing concept semantic web. The semantic web is an extension to existing web standards that enables semantic information to be associated with web documents. The current World Wide Web is not designed to be easily understandable by machines. The main objective of the semantic web is to make web documents easier for machines to understand. It proposes to add machine-readable information to the documents. However, the vast majority of current web pages have no semantic information associated with them. One of the main issues of the semantic web is the difficulty in adding semantic tags to large amounts of text. The ability to automatically add semantic annotations would be of huge benefit to adoption of the semantic web. IE is one process that can be used for automatically identifying entities in existing web documents and using this information to add semantic annotations to the documents.

[1]MEDLINE (Medical Literature Analysis and Retrieval System Online) is a bibliographic database of life sciences and biomedical information owned by the United States National Library of Medicine (NLM). MEDLINE is freely available on the Internet and searchable via PubMed: http://www.ncbi.nlm.nih.gov/entrez/query.fcgi

While IE spans a wide range of application areas, we anticipate that there will be even more in the near future. Particularly, as speech-understanding technology improves, the need for IE capabilities will increase dramatically. The need for the information is unending and the role of the language in exchanging and disseminating the information is indisputable, and therein lies the future of IE applications.

1.2 Thesis Statement

The main objective of the study is to build a system that automatically locates and extracts information from Turkish unstructured texts. Our study focuses on two basic IE tasks: named entity recognition and entity relation detection. Adopting supervised learning strategy, the developed IE system automatically starts with a set of examples collected from a training dataset and generates the extraction rules from the given examples by using a carefully designed learning strategy. Since the system does not rely on handcrafted rules/patterns, it does not heavily suffer from domain adaptability problem. Moreoever, an adapted version of the automatic rule learning method is experimented for protein name extraction task. Besides a novel rule learning algorithm, our system employs rule filtering and rule refinement techniques to minimize any possible reduction in accuracy caused by the generalization. In order to obtain accurate generalization and remedy the issues related to the data sparseness problem, the developed IE system uses an expressive rule representation language and several syntactic and semantic features of the text, including: orthographical, contextual, lexical and morphological features. In particular, morphological features of the text are effectively used in this study to increase the extraction performance for Turkish, an agglutinative language that is therefore morphologically rich and productive. Because of the lack of defined task definitions and training data for Turkish, this study covers the adaptation of the NER and ERD task definitions to Turkish and the development of an annotated corpus.

1.3 Organization of the Dissertation

The structure of the thesis is as follows. Chapter 2 reviews the previous research in IE field. Chapter 3 provides a foundation for further chapters. Chapter 4 (based on [94]) and 5 describe how we employed automatic rule learning for the tasks of NER and ERD respectively. Chapter 6 presents the experimental evaluation of the study. Finally, in the last chapter we conclude and indicate directions for future research.

Chapter 2

Related Work

IE has been well-researched and many approaches have been proposed ranging from handcrafted rule-based systems to adaptive learning systems. Numerous studies [60, 89, 102, 72] have reviewed the studies that have been carried out by the research community in the field of IE. Kushmerick and Thomas [60] focused on machine learning approaches for IE. They segmented the field of adaptive IE roughly into two areas: finite state techniques that learn extraction knowledge corresponding to regular grammars or automata, and the relational rule learning techniques that learn first-order Prolog-like extraction rules. Siefkes and Siniakov [89] surveyed the adaptive IE systems and established a classification of different types of adaptive IE systems based on their observations on the origins and requirements. Turmo et al. [102] compared different adaptive IE approaches that use machine learning techniques used to achieve adaptive IE technology. In their own survey, Nadeau and Sekine [72] reviewed the research conducted in the Named Entity Recognition and Classification (NERC) field between 1991 and 2006. In addition to the different techniques proposed in the field, they reported their observations about languages, named entity types, domains and textual genre studied in the literature.

2.1 The Message Understanding Conferences (MUCs)

In order to stimulate the development of new IE systems and to create a common basis for the evaluation of their performance, several projects were established. The Message Understanding Conferences (MUCs) [21, 5, 2, 1, 4, 3], a series of seven conferences held between 1987 and 1998, were a great spur to research in the field. MUC funded the development of metrics and algorithms to support evaluations of emerging information extraction technologies by providing a platform on which various IE approaches can be evaluated and compared. In each evaluation, task, training data, test data and a scoring metric were provided to participants.

The tasks grew from just production of a database of events found in newswire articles from one source to the production of multiple databases of increasingly complex information extracted from multiple sources of news in multiple languages. Named Entity Recognition (locating the entities), Coreference Resolution (finding identities between entities), Template Element Construction (finding the attributes of the entities), Template Relation Construction (detecting relationships between entities), and Scenario Template Construction (extracting events and identifying event-attributes) are the major tasks defined during the MUCs.

The results of MUC evaluations were reported at conferences during the 1990's where developers and evaluators shared their findings and government specialists described their needs. Table 2.1 lists the year and topics (domains) of each evaluation.

Many new problems were identified and separated. During the evaluations, evaluation metrics and methods were determined. Moreover, many corpora with associated "key templates" were developed. The only downside of the evaluations may be that the participating systems tended to converge to a few best performing approaches due to the competitive nature of the evaluations. The MUC program

was finalized after MUC-7 because of the funding problems. A brief history of MUC evaluations was provided by Grishman and Sundheim [39].

Project	Year	Domain
MUC-1	1987	Naval operations messages
MUC-2	1989	Naval operations messages
MUC-3	1991	Terrorism in Latin American countries
MUC-4	1992	Terrorism in Latin American countries
MUC-5	1993	Corporate Joint Venture and Microelectronics
MUC-6	1995	News articles on management changes
MUC-7	1998	Airplane Crashes/Rocket Launches

Table 2.1: List of MUC Evaluations

2.2 Automatic Content Extraction (ACE) Program

The Automatic Content Extraction (ACE) [73] evaluation program, a successor to the MUCs, began in 1999 with the aim of developing automatic content extraction technology to support automatic processing of human language in text form from a variety of sources.

The ACE evaluations largely follow the scheme of the MUCs. Its development cycle includes specifying the tasks, developing training and test data, carrying out an evaluation and discussing the results from all participating sites. Several tasks were defined during the evaluations:

- Entity Detection and Tracking (EDT): detecting each unique entity mentioned in the source text, and tracking its mentions.
- Relation Detection and Characterization (RDT): detecting and characterizing relations between EDT entities.
- Entity Detection and Recognition (EDR): the detection of the entities, recognition of the information about the detected entities and creating a unified representation for each entity.

- Relation Detection and Recognition (RDR): the detection of the relations, recognition of the information about the detected relations and creating a unified representation for each relation.
- Time Detection and Recognition(TDR): detecting and recognizing the temporal expressions mentioned in the text.
- Value Detection and Recognition (VDR): the detection of the values (e.g.money, contact-info), recognition of the information about the detected values and creating a unified representation for each value.
- Event Detection and Recognition: the detection of the events, recognition of the information about the detected events and creating a unified representation for each event.
- Local Entity Detection and Recognition (LEDR): the detection of the entities in each document in a document collection separately, recognition of the information about the detected entities and creating a unified representation for each entity.
- Local Relation Detection and Recognition (LRDR): the detection of the relations in each document in a document collection separately, recognition of the information about the detected relations and creating a unified representation for each relation.
- Global Entity Detection and Recognition (GEDR): the detection of the entities in a document collection collectively, recognition of the information about the detected entities and creating a unified representation for each entity.
- Global Relation Detection and Recognition (GRDR): the detection of the relations in a document collection collectively, recognition of the information about the detected relations and creating a unified representation for each relation.

One difference from the MUC evaluations is that it is multi-source and multilingual. Each evaluation includes text from different sources; e.g. newswire

documents, broadcast news transcripts, and text derived from OCR. ACE Evaluations also cover several languages: English, Chinese, Arabic, and Spanish. Table 2.2 lists the tasks and languages of the ACE evaluations.

After several evaluations took place between 1999 and 2008 in order to accomplish this goal, ACE became a track in the Text Analysis Conference (TAC) [74] in 2009.

Tasks	**Languages**	**Tasks**
2000	English	EDT (Pilot)
2001	English	EDT, RDC
2002	English	EDT, RDC
2003	English, Chinese, Arabic	EDT, RDC
2004	English, Chinese, Arabic	EDR, RDR, TDR
2005	English, Chinese, Arabic	EDR, RDR, TDR, VDR, Event DR
2007	English, Chinese, Arabic, Spanish	EDR, RDR, TDR, VDR, Event DR
2008	English, Arabic	LEDR, LRDR, GEDR, GRDR

Table 2.2: List of ACE Evaluations

2.3 Approaches and Methods

In this section, we will cover the IE approaches and methods result of previous research. In fact, the idea is not a new one. The information extraction concept was first introduced by Harris [42] in the 1950's. First applications [46, 84] were reported within the medical domain. Furthermore, the task of automatically extract information from natural language texts has received a lot of attention in the past, and as such we observe a high diversity in the proposed approaches and the methods used therein.

We will follow the general trend of natural language technology, which is a transition from complete human intervention to automated optimization, to introduce the proposed methods in the past. Early research [7, 37, 51] in the IE community established a linguistic architecture based on cascading automata

and domain specific knowledge. The SRI FASTUS system [7] used a series of finite-state transducers that compute the transformation of text from sequences of characters to domain templates. The Proteus system [37] also used cascaded finite state transducers to recognize succession events. At a low syntactic level, transducers were prepared to locate proper names, noun groups and verb groups; at a higher syntactic and semantic level, transducers were generated to account for basic events. The LaSIE-II system [51], developed at the University of Sheffield, used finite state recognition of domain-specific lexical patterns, partial parsing using a restricted context-free grammar and quasi-logical form (QLF) representation of sentence semantics. Although these systems have demonstrated remarkable performance, rule development and management is the main issue in these systems. Developing and managing rules by hand requires high human expertise. Constructing IE systems manually has also proven to be expensive [81]. Domain adaptability is also a major issue for these systems since the domain specific rules constructed in these systems for a domain cannot be easily applied to another domain.

In order to reduce human effort in building or shifting an IE system, significant research in information extraction has focused on using supervised learning techniques for automated development of IE systems. Instead of having humans create patterns and rules, these models use automatically generated rules via generalization of examples or statistical models derived from the training data. One of the earliest systems, AutoSlog [80] learns a dictionary of patterns, called concept nodes, with an anchor word, most often the head verb, to activate that concept node to extract information from text. The LIEP system [50] is a learning system that generates multi-slot extraction rules. The CRYSTAL system [92] employed inductive learning to construct a concept dictionary from annotated training data. Inspired by inductive logic programming methods, RAPIER [14, 15] used bottom-up (specific to general) relational learning to generate symbolic rules for IE. Freitag [30] describes several learning approaches to the IE problem: a rote learner, a term-space learner based on Naive Bayes, an approach using grammatical induction, and a relational rule learner. Freitag also proposed a multi-strategy approach which combines the described learning

approaches. Basically, wrappers can be seen as simple extraction procedures for semi-structured or highly structured data. Freitag and Kushmerick [31] introduced wrapper induction, identified a family of six wrapper classes, and demonstrated that the wrappers were both relatively expressive, and efficient for extracting information from highly regular documents. Hsu and Dung [48] presented SoftMealy, a wrapper representation formalism based on a finite state transducer and contextual rules. The Boosted Wrapper Induction (BWI) method [31, 59] learns a large number of relatively simple wrapper patterns, and combines them using boosting. The Hidden Markov Models (HMMs) are powerful statistical models that have been successfully applied to the task of information extraction [12, 33, 32, 87]. One of the earliest learning systems for IE based on HMMs is the IdentiFinder system developed by Bikel et al. [12]. Freitag and McCallum [33] used shrinkage to improve parameter estimation of the HMM emission probabilities and learn optimal HMM structures. Seymore et al. [87] focused on learning the structure of the HMMs. Maximum entropy Markov model (MEMM) [66], Conditional Random Fields (CRFs) [67, 76], Maximum entropy models [16], and Support Vector Machines (SVMs) [27, 108] were also used for information extraction.

The adaptive methods discussed thus far used supervised learning strategy. Supervised methods can quickly learn the most common patterns, but require a large corpus in order to achieve good coverage of the less frequent patterns. However, annotating a large corpus is not easy. Semi-supervised (or weakly supervised) methods have been developed to overcome the annotated corpus preparation problem. Because, the amount of un-annotated text is greater than the annotated data, semi-supervised methods use un-annotated text along with a small set of annotated data. The major technique in this category is called "bootstrapping". Bootstrapping methods [82, 105, 22] use only a small degree of supervision, such as a set of seeds, at the beginning. Riloff and Jones [82] introduced a multi-level bootstrapping technique. They used mutual bootstrapping technique that learns extraction patterns from the seed words and then exploits the learned extraction patterns to identify more words that belong to the semantic category. To minimize the system's sensitivity to noise,

they introduced another level of bootstrapping (meta-bootstrapping) that retains only the most reliable lexicon entries produced by mutual bootstrapping and then restarts the process. A different solution approach to the annotated corpus preparation problem is to mark only the data which can help to improve the overall accuracy. Active learning methods [71, 97, 53] try to make this process by selecting suitable candidates for the user to annotate. Thompson et al. [97] showed that 44% example savings can be achieved by employing active sample selection. The methods based on unsupervised learning approaches [6, 88, 26] do not need labeled data at all. Shinyama and Sekine [88] used the time series distribution of words in news articles to obtain rare NEs. KnowItAll system [26] uses a set of generic extraction patterns, and automatically instantiates rules by combining these patterns with user supplied relation labels.

2.3.1 Review of the previous IE Systems

After reviewing the general approaches to IE task, we believe that it is helpful to examine some important works in detail in the following sections.

2.3.1.1 FASTUS

The FASTUS system [47, 7] used an architecture consisting of cascaded finite state transducers, each providing an additional level of analysis of the input, together with merging of the final results. The system employed six transducers.The first transducer, the Tokenizer, accepts a stream of characters as input, and transforms it into a sequence of tokens. Next, the Multiword Analyzer automatically recognizes token sequences (like "because of") that are combined to form single lexical items. The Preprocessor handles more complex or productive multiword constructs than could be handled automatically from the lexicon. Named entities are recognized by the Name Recognizer. It also locates unknown words and sequences of capitalized words that don't fit other known name patterns, and flags them so that subsequent transducers can determine their type, using broader context. Next comes the Parser where noun groups and verb groups

are output. The Combiner produces larger constituents (e.g. "John Smith, 56, president of Foobarco"), from the output of the parser. The final transducer, the Domain, recognizes the particular combinations of subjects, verbs, and objects that are necessary for correctly filling the templates for a given information extraction task. The FASTUS system also includes a merger for merging, a unification operation, the templates produced by the domain phase. The precise specifications for merging are provided by the system developer when the domain template is defined.

2.3.1.2 Proteus

The Proteus system [37] also used cascaded finite state transducers to perform IE tasks. In a similar fashion to the FASTUS system, the Proteus system performs text analysis in seven main stages: (1) tokenization and dictionary look-up, (2) name recognition, (3) noun group recognition, (4) verb group recognition, (5) semantic pattern recognition, (6) reference resolution, and (7) response generation.

In the first stage, the input document is divided into tokens and each token is looked up in our dictionaries. This initial stage is followed by four pattern matching stages. The name recognition stage records the initial mention and type of each name. The second pattern matching stage, noun group recognition, recognizes noun groups (i.e. nouns with their left modifier). Next, both active and passive verb groups are found. During the semantic pattern recognition stage, the scenario-specific patterns are recognized. The various stages of pattern matching produce a logical form for the sentence, consisting of a set of entities and a set of events which refer to these entities. Reference resolution examines each entity and event in logical form and decides whether it is an anaphoric reference to a prior entity or event, or whether it is new and must be added to the discourse representation. Finally, response generation handles the required inferencing for generating the results for several IE tasks.

2.3.1.3 LaSIE-II

The LaSIE-II system [51] is a pipeline of modules each of which processes the entire text before the next is invoked. The system starts with basic preprocessing operations: tokenization, gazetteer look-up, sentence splitting, part-of-speech tagging, and morphological analysis. The text processing continues with partial parsing using a restricted context-free grammar and quasi-logical form (QLF) representation of sentence semantics. The parsing results of sentences are mapped to QLF representation. Then, the discourse interpreter adds the QLF representation to a semantic net. This semantic map keeps the system's domain model as a hierarchy of concepts. Additional information gathered is also added to the model, then coreference resolution is performed, and finally information consequent upon the input is added. This results in an updated discourse model. Lastly, the template writer generates the results for different IE tasks by scanning the discourse model and extracting the required information.

2.3.1.4 AutoSlog

AutoSlog [80] automatically constructs a domain-specific dictionary for information extraction. Using supervised learning strategy, given a set of training texts and their associated answer keys, AutoSlog learns a dictionary of patterns that are capable of extracting the information in the answer keys from the texts. These patterns are called concept nodes. A concept node is essentially a case frame that is triggered by a lexical item, called conceptual anchor point, and activated in a specific linguistic context. AutoSlog provides 13 single slot predefined concept node types to recognize a specific linguistic pattern. An example concept node is

$$<subject>\ passive\text{-}verb$$

with an anchor point murdered. This concept node was generated by the system given the training clause "the diplomat was murdered" along with "the diplomat" as the target string. Since the target string is the subject of the training clause and is followed by a passive verb "murdered", the system proposed a concept

node type that recognizes the pattern <*subject*> *passive-verb* is satisfied. The concept node type returns the word "murdered" as the conceptual anchor point along with enabling conditions that require a passive construction.

2.3.1.5 PALKA

The PALKA system [56] automatically acquires extraction patterns that are in the form of frame-phrasal pattern structures (FP-structures) from a training corpus. An FP-structure is a pair of a meaning frame and a phrasal pattern. Each slot in the meaning frame defines an item-to-be-extracted together with the semantic constraints associated to it (e.g. the target of the bombing event must be a physical object). The phrasal pattern represents an ordered sequence of lexical entries and/or semantic categories taken from a predefined concept hierarchy. The frame-phrasal pattern representation in the PALKA system is shown in Figure 2.1.

```
- The meaning frame
  (BOMBING
    agent: ANIMATE
    target: PHYSICAL-OBJ
    instrument: PHYSICAL-OBJ
    effect: STATE
  )

- The phrasal frame
  ((BOMB) EXPLODED AT (PHYSICAL-OBJ))

- The FP-structure
  (BOMBING
    target: PHYSICAL-OBJ
    instrument: BOMB
    pattern: ((instrument) EXPLODED AT (target))
  )
```

Figure 2.1: The frame-phrasal pattern representation in the PALKA system

The FP-structures are used by the parser of the system to extract the relevant information resident in the input texts. Applying FP-structures to input texts

happens in two steps: (1) An FP-structure is activated when a phrase in the input text is matched to the elements in a phrasal pattern, and (2) The relevant information is extracted by using the activated meaning frame.

2.3.1.6 WHISK

The WHISK system [91] is a supervised learning system that generates extraction rules. The WHISK extraction patterns are a special type of regular expressions that can represent the context that makes a phrase relevant, and the exact delimiters of the phrase. WHISK patterns can be used on both semi-structured and free text domains. Depending on the structure of the text, WHISK can generate patterns that use either context-based representation or delimeter-based representation or both. An example WHISK rule is shown in Figure 2.2. The rule implies: (1) skip until the first digit followed by the "br" string; extract the recognized digit into the "Bedrooms" slot in the target template. (2) skip until a dollar sign immediately followed by a number; extract the recognized number into the "Price" slot in the target template.

- Extraction Rule
ID:: 1
Pattern:: * (Digit) BR * $ (Number)
Output:: Rental {Bedrooms $1} {Price $2}

- Input Text
Capitol Hill - 1 br twnhme. fplc D/W W/D.Undrgrnd pkg incl $675. 3 BR, upper flr turn of ctry HOME. incl gar, grt N. Hill loc $995. (206) 999-9999

- Extracted Info
RENTAL **{** BEDROOM = "1"; PRICE = "$675";**}**
RENTAL **{** BEDROOM = "3"; PRICE = "$995";**}**

Figure 2.2: An extraction rule in the WHISK system

2.3.1.7 CRYSTAL

The CRYSTAL system [92] automatically induces extraction rules from annotated training data in order to extract relevant information from texts. These rules, called concept definitions, use a combination of syntactic, semantic, and lexical constraints to identify references to the target concept.

CRYSTAL uses supervised learning strategy with a bottom up approach, which begins with highly specific concept definitions and tries to merge similar concept nodes by gradually relaxing the constraints. The merged concept is tested on the training data and error rate for the new concept is calculated. If the found error rate exceeds an error tolerance parameter, CRYSTAL begins the generalization process on another initial CN definition. This process continues until no unification can be executed. The error tolerance parameter can be used to make the learning process robust. The parameter also determines the trade-off between precision and recall of the learned patterns.

2.3.1.8 RAPIER

The RAPIER system [14, 15] uses a generalization technique inspired by Inductive Logic Programming (ILP) to generate symbolic rules for extraction. The RAPIER extraction rules are indexed by template name and slot name and contains three parts: 1) a pre-filler pattern (matches text immediately preceding the target field), 2) a pattern (matches the target field), and 3) a post-filler pattern (matches the text immediately following the target field).

RAPIER's learning strategy is compression-based and employs a specific to general (bottom-up) search. The generalization algorithm generates more general rules by selecting several random pairs of rules from the rulebase, finding generalizations of the selcted rule pairs, and selecting the best rule among the acceptable rules to add to the rulebase. The old rules covered by the newly added rule (i.e. the ones which cover a subset of the examples covered by the new rule) are removed from the rulebase when the new rule is added to the rulebase.

2.3.1.9 SRV

The SRV system [29] is based on a top-down relational algorithm. The system treats information extraction is a kind of text classification, where every candidate instance in a document is presented to a classifier, which is asked to accept or reject them as target information field to extract.

The SRV system provides two basic types of generalization features: simple and relational. A simple feature (e.g. numeric, capitalized, verb) is a function mapping a token to some discrete value. On the other hand, a relational feature (e.g. prev_token, next_token) maps a token to another token.

SRV starts learning with the entire set of examples (i.e. all negative examples and any positive examples not covered by already induced rules) and adds predicates greedily, attempting to maximize the number of positive examples covered, while minimizing the number of negative examples covered. SRV validates each learned rule on a hold-out set, a randomly selected portion of the training data. After training on the remaining data, the number of matches and correct predictions over the validation set is stored with each rule. This validation scores are used during testing in order to calculate system's prediction confidence.

2.3.1.10 Boosted Wrapper Induction

Boosted Wrapper Induction (BWI) method [31] learns a large number of simple extraction procedures (called wrappers) and combines them using boosting which is a method for improving the performance of a simple machine learning algorithm by repeatedly applying it to the training set. BWI treats IE as a token classification task, where the task is to classify each token as being a boundary that marks the beginning or end of a field. It learns two separate models: one for detecting the start boundaries and one for detecting the end boundaries. When start and end boundaries are detected, a phrase is extracted based on the probability of a target phrase of that length occurring.

2.3.2 Domains

From security to medical field, possible application areas of the research include a variety of domains. MUC-3 [5] and MUC-4 [2] evaluations performed on the reports of terrorist events in Central and South America, as reported in articles provided by the Foreign Broadcast Information Service. Louis et al. [63] applied IE technology to cyber forensic domain and introduced a probabilistic NER system for the identification of names in documents for the purpose of forensic investigation. Maynard and colleagues [65] developed a system that can identify named entities in diverse text types such as emails, scientific documents and religious text. Minkov et al. [69] investigated IE for informal text with an experimental study of the problem of recognizing personal names in emails. Wellner et al. [103] conducted their experiments on a data set of research paper citations. As part of their study to minimize human interaction during corporate expense reimbursement process, Zhu et al. [109] presented a CRF based approach for extracting relevant named entities from document images. ProMED-PLUS system [106] can automatically extract the facts from plain text reports about outbreaks of infectious epidemics around the world. Since most of the information on the World Wide Web is in textual format, various studies [48, 58, 70, 29, 91, 8, 25, 49] have increasingly been conducted to extract information from the Web. The biomedical domain is one of the domains that many IE studies have focused on. One of the basic tasks in automatic extraction of information from biological texts is protein name recognition. Tsuruoka and Tsujii [100] proposed using approximate string searching techniques and expanding the dictionary in advance with a probabilistic variant generator for protein name recognition. Fukuda et al. [36] developed PROPER (PROtein Proper-noun phrase Extracting Rules) system that exploits simple lexical patterns and orthographic features for protein name recognition. Franzén et al. [28] introduced the YAPEX system that combines lexical and syntactic knowledge, heuristic rules and a document-local dynamic dictionary. Tanabe and Wilbur [93] proposed ABGene system which uses both statistical and knowledge-based strategies for finding gene and protein names. NLProt [68] is a system that combines a preprocessing dictionary and rule based filtering step with several

separately trained support vector machines to identify protein names in the MEDLINE abstracts. Tatar and Cicekli [95] introduced two different techniques -a statistical learning method based on bigram language model and an automated rule learning method- along with the hierarchically categorized syntactic token types to identify protein names located in the biological texts.

2.3.3 Languages

Although the language subject to most research applications is English, there has been a growing attention to other languages. The shared task of CoNLL-2002 [98] focused on NER for Spanish and Dutch. A year later, German was one of the focus languages in CONLL-2003 [99]. Numerous studies [104, 107, 35] have been conducted on IE in Chinese. Japanese has received a lot of attention as well [86, 52]. Moreover, various studies deal with the development of systems for addressing IE in various languages: Korean [19], French [77], Greek [77, 13], Danish [11], Italian [22], Vietnamese [96], Bengali [43], Arabic [10], Bulgarian [90], Russian [78], and Ukrainian [54]. Multilingual IE has also received a lot of attention [40, 85]. Cucerzan and Yarowsky [23] presented a language-independent bootstrapping algorithm and conducted their experiments on several languages: Romanian, English, Greek, Turkish and Hindi. This was the first study conducted to examine named entity recognition in Turkish to our knowledge, an otherwise seldom researched language. Tur et al. [101] applied statistical learning approaches to a number of tasks for Turkish: sentence segmentation, topic segmentation, and name tagging. Their named tagging approach is based on n-gram language models embedded in hidden Markov models. Bayraktar and Temizel [9] studied *Person* name extraction from Turkish financial news articles using local grammar approach. Conducting their experimentation on different text genres (news articles, historical text, and child stories), Kucuk and Yazici [57] presented a rule based named entity recognition system for Turkish which employs a set of lexical resources and pattern bases for the extraction of named entities.

Chapter 3

Preliminaries

The objective of this chapter is to provide the necessary foundation for the next two chapters where we present the details of the technique we propose for IE. The following section covers the basic knowledge of Turkish and the IE related challenges drawn from the nature of the language. Lastly in this chapter, the concept of *Specific Generalization of Strings* is briefly described. Originally proposed in order to reduce over-generalization problem in the learning process of predicates with string arguments, this ILP technique is adapted for IE in our study.

3.1 Turkish

Turkish is a member of the Oghuz group of the Turkic languages, which belongs to the Altaic branch of Ural-Altaic language family. Turkish uses a Latin alphabet consisting of twenty-nine letters, of which eight are vowels and twenty-one are consonants. Similar to Hungarian and Finnish, Turkish has vowel harmony and lacks grammatical gender distinction.

Another major feature of Turkish is that it is an agglutinative language with free word order [75]. The complex morphological structure of Turkish words have

a significant role to play in IE; it makes the task even more difficult. In Turkish, a sequence of inflectional and derivational morphemes can be added to a word. This concatenation process can yield relatively long words, which can convey the equivalent meaning of a phrase, or even a whole sentence in English. A single Turkish word can give rise to a very large number of variants, which results in the vocabulary explosion.

Surface Form	**Morphological Decomposition**	**English Meaning**
İstanbul	istanbul +Noun +Prop +A3sg +Pnon +Nom	İstanbul
İstanbul'da	istanbul +Noun +Prop +A3sg +Pnon +Loc	in İstanbul
İstanbul'daki	istanbul +Noun +Prop +A3sg +Pnon +Loc ^DB+Adj+Rel	the (one) in İstanbul
İstanbul'dakiler	istanbul +Noun +Prop +A3sg +Pnon +Loc ^DB+Adj+Rel ^DB +Noun+Zero+A3pl+Pnon+Nom	the ones in İstanbul
İstanbul'dakilerden	istanbul +Noun +Prop +A3sg +Pnon +Loc ^DB+Adj+Rel ^DB +Noun+Zero+A3pl+Pnon+Abl	from the ones in İstanbul

Table 3.1: Several surface forms produced using the stem word İstanbul

+Noun ⇒ Noun; +Prop ⇒ Proper Noun ; +Pnon ⇒ Pronoun (no overt agreement); +A3sg ⇒ 3rd person singular; +A3pl ⇒ 3rd person plural; +Nom ⇒ Nominative; +Loc ⇒ Locative; +Abl ⇒ Ablative; ^DB+Adj+Rel ⇒ Derivation Boundary + Adjective + Relative; ^DB+Noun+Zero ⇒ Derivation Boundary +Noun + 0 Morpheme;

Table 3.1 lists several formations produced using the stem word *İstanbul*. Note that the morphemes added to the stem word produce different surface forms. The list can easily be expanded; (e.g. *İstanbul'dakilerdenmiş*, *İstanbul'dakilerdenmişce*,...). In fact, millions of different surface forms can be derived from a nominal or verbal stem [41]. Although in English, it is possible that a suffix can change the surface form of a proper noun (e.g. Richard's), it is not as common as in Turkish and other morphologically rich languages. Using each surface form generated from the same stem as a different training element would cause data sparseness problem in the training data, which indicates that

morphological level processing is a requirement for Turkish IE.

3.2 Specific Generalization of Strings

Specific generalization of strings, described in [20], is based on an observation that humans learn general sentence patterns using similarities and differences between many different example sentences that they are exposed to. The basic idea behind the concept is to generalize the strings by processing similarities and differences between them. A similarity (SIM) represents a similar part between two strings, and a difference ($DIFF$) represents a pair of differing parts between two strings. The similarities and the differences are the basic elements of a match sequence (MS) which is defined as the sequence of similarities and differences between two strings with certain conditions satisfied. For instance, a similarity cannot follow another similarity, and a difference cannot follow another difference in a match sequence. The conditions for the match sequence is important because they guarantee that there can be at least one match sequence for any given two strings. However, conditions cannot provide that there can be at most one match sequence for any given two strings.

A specific case of a match sequence, unique match sequence (UMS), can be described as a match sequence which can occur either uniquely once or none at all for any given two strings. To meet these criteria, the notion of unique match sequence has two more necessary conditions on a match sequence. The first condition states that a symbol cannot occur in any difference, if it occurs in a similarity. Moreover, the second condition says that a symbol cannot occur in the second constituent of any difference if the same symbol is found in the first constituent of a difference. The examples provided below will clarify the unique match sequence concept.

- $UMS(\epsilon, \epsilon) = SIM(\epsilon)$
- $UMS(\text{ab},\text{ab}) = SIM(\text{ab})$

- *UMS*(bc, ef)= *DIFF*(bc, ef)
- *UMS*(abcb, dbebf)= *DIFF* (a, d) *SIM*(b) *DIFF* (c, e) *SIM*(b) *DIFF* (ϵ, f)
- *UMS*(abb, cdb)= $\varnothing$
- *UMS*(ab, ba)= $\varnothing$

As evident from the examples, the unique match sequence of two empty strings is a sequence of a single similarity which is an empty string. Moreover, the unique match sequence of two identical strings is a sequence of a single similarity which is equal to that string. The unique match sequence of two totally different strings is a sequence of a single difference.

In the framework, the separable difference (*SDIFF*) term is coined to provide further capturing of similar patterns and to avoid the ambiguity. A difference $DIFF(D_1, D_2)$ is said to be separable by difference $DIFF(d_1, d_2)$ if d_1 and d_2 occur more than once and the same number of times in D_1 and D_2, respectively. A difference D is said to be useful separation difference for a match sequence (or an instance of match sequence) if all the differences in that match sequence are separable by D, and the total number of differences which occur more than once is increased after the separation. The next definition is the most useful separation difference (*MUSDIFF*)which is the separation difference that separates the match sequence with the greatest factor.

In [20], an algorithm which can find the specific generalization of two strings presented. In the algorithm, the specific instance of a unique match sequence (*SIofUMS*) is computed by dividing the unique match sequence iteratively by the most useful separation difference. The algorithm replaces all differences in the found specific instance of the match sequence with new variables replace the same differences with the same variables in order to create the specific generalization (*SG*)of two strings. The example below shows the generation of the specific generalization of two strings:

- *UMS*(abcdbhec,agcdgfec) = a (b, g) cd (bh, gf) ec

- $MUSDIFF$(abcdbhec,agcdgfec) = (b, g)
- $SIofUMS$(abcdbhec,agcdgfec) = a (b, g) cd (b, g) (h, f) ec
- SG(abcdbhec,agcdgfec) = aXcdXYec

Chapter 4

Named Entity Recognition

4.1 Task Definition

Named Entity Recognition, finding named entities (persons, locations, organizations, etc.) located in unstructured texts, is one of the most fundamental IE tasks. NER is a prerequisite to more sophisticated information extraction tasks, such as entity relation detection and event extraction. The main objective of the task is to extract and categorize all instances of predetermined categories of names and certain expressions in a given set of documents.

In this study, we generally followed the existing MUC-7 named entity task definition [17] as a guideline. However, particular characteristics of the Turkish language and the scope of our study required us to make some adaptations to the task definiton specified in the guideline. Of the three subtasks (*entity names*, *temporal expressions*, and *numerical expressions*) defined in [17], leaving *numerical expressions* out, we only included two of them in our task definiton. Moreover, morphemes coming after the named entities are considered as part of the name in our study. The following subsections describe details of our task definiton.

4.1.1 Scope

The *entity names* subtask is limited to proper names, acronyms, and sometimes miscellaneous other unique identifiers, which are categorized as follows:

- *PERSON*: named person or family.
- *ORGANIZATION*: corporate, governmental, or other organizational entity.
- *LOCATION*: name of politically or geographically defined location (e.g. cities, provinces, countries, international regions, bodies of water, mountains).

The *temporal expressions* subtask is for "absolute" and "relative" temporal expressions and categorized as follows:

- *DATE*: complete or partial date expression.
- *TIME*: complete or partial expression of time of day.

4.1.2 General Guidelines

- Morphemes coming after the named entities are considered as part of the name.

Input Text: *"Kamuran Mustafa Ballı'nın"*
Extraction: *<PERSON> Kamuran Mustafa Ballı'nın </PERSON>*

- Conjoined named entities *in general* are to be extracted separately.

Input Text: *"İsmail Bulat ve Mustafa Erdoğan"* (*İsmail Bulat and Mustafa Erdoğan*)
Extraction: *<PERSON> İsmail Bulat </PERSON> ve <PERSON> Mustafa Erdoğan </PERSON>*

- Conjoined multi-name expressions are to be extracted as single entities.

Input Text: *"Muhterem ve Güleser Şahin"*
Extraction: *<PERSON> Muhterem ve Güleser Şahin </PERSON>*

Input Text: *"Balat ve Tarabya Polis Karakolları"* (*Balat and Tarabya Police Stations*)
Extraction: *<ORGANIZATION> Balat ve Tarabya Polis Karakolları </ORGANIZATION>*

- Single-name expressions containing conjoined modifiers with no elision are to be extracted as single entities.

Input Text: *"Hakkari Dağ ve Komando Tugayı"* (*Hakkari Mountain and Commando Brigade*)
Extraction: *<ORGANIZATION> Hakkari Dağ ve Komando Tugayı </ORGANIZATION>*

Input Text: *"Taksim İlkyardım Eğitim ve Araştırma Hastanesi"* (*Taksim First Aid Training and Research Hospital*)
Extraction: *<ORGANIZATION> Taksim İlkyardım Eğitim ve Araştırma Hastanesi </ORGANIZATION>*

- In case of nested expressions, only outer expressions are to be extracted.

Input Text: *"5 Ocak Caddesi"* (*5th January Street*)
Extraction: *<LOCATION> 5 Ocak Caddesi </LOCATION>*

4.1.3 Organization Names

- Miscellaneous types of proper names that are to be extracted as *ORGANIZATION* include stock exchanges, multinational organizations, political parties, orchestras, unions, non-generic governmental entity names, sports teams and armies.

Input Text: *"TBMM"* (*acronym for Turkish Grand National Assembly/Parliament*)
Extraction: *<ORGANIZATION> TBMM </ORGANIZATION>*

Input Text: *"Türk Silahlı Kuvvetleri"* (*Turkish Armed Forces*)
Extraction: *<ORGANIZATION> Türk Silahlı Kuvvetleri </ORGANIZATION>*

Input Text: *"Türk Bankacılar Birliği"* (*The Banks Association of Turkey*)
Extraction: *<ORGANIZATION> Türk Bankacılar Birliği </ORGANIZATION>*

Input Text: *"Ziraat Bankası"* (*Ziraat Bank*)
Extraction: *<ORGANIZATION> Ziraat Bankası </ORGANIZATION>*

- Miscellaneous types of proper names referring to facilities (e.g., mosques, churches, embassies, factories, hospitals, hotels, museums, universities) are to be extracted as *ORGANIZATION*.

Input Text: *"İpekyolu Cami"* (*Ipekyolu Mosque*)
Extraction: *<ORGANIZATION> İpekyolu Cami </ORGANIZATION>*

Input Text: *"Aşkale Çimento Fabrikası"* (*Askale Cement Factory*)
Extraction: *<ORGANIZATION> Aşkale Çimento Fabrikası </ORGANIZATION>*

Input Text: *"Betyaakov Sinagogu"* (*Betyaakov Synagog*)
Extraction: *<ORGANIZATION> Betyaakov Sinagogu </ORGANIZATION>*

Input Text: *"Van 100. Yıl Üniversitesi"* (*Van 100th Year University*)
Extraction: *<ORGANIZATION> Van 100. Yıl Üniversitesi </ORGANIZATION>*

- Generic entity names such as "the police" and "the government," are not to be extracted.

Input Text: *"Jandarma"* (*the gendarmerie*)
Extraction: *None*

Input Text: *"Emniyet"* (*the police*)
Extraction: *None*

4.1.4 Person Names

- Named person or families are to be extracted as *PERSON*.

Input Text: *"Mustafa Yücel Özbilgin"*
Extraction: *<PERSON> Mustafa Yücel Özbilgin </PERSON>*

Input Text: *"M.Cahit Kıraç"*
Extraction: *<PERSON> M.Cahit Kıraç </PERSON>*

- Titles such as "Dr." and role names such as "President" are not considered part of a person name.

Input Text: *"Cumhurbaşkanı Ahmet Necdet Sezer"* (*President Ahmet Necdet Sezer*)
Extraction: *Cumhurbaşkanı <PERSON> Ahmet Necdet Sezer </PERSON>*

Input Text: *"Org.Hilmi Özkök"* (*GEN Hilmi Özkök*)
Extraction: *Org. <PERSON> Hilmi Özkök </PERSON>*

- Partial Names (names without first names, or names without family names) are to be extracted as *PERSON*.

Input Text: *"Erdoğan"*
Extraction: *<PERSON> Erdoğan </PERSON>*

Input Text: *"Demirel"*
Extraction: *<PERSON> Demirel </PERSON>*

- Names not clearly mentioned for basic legal and ethical reasons are to be marked as *PERSON*.

Input Text: *"T.F."*
Extraction: *<PERSON> T.F. </PERSON>*

Input Text: *"Ahmet K."*
Extraction: *<PERSON> Ahmet K. </PERSON>*

4.1.5 Location Names

- Miscellaneous types of proper names that are to be extracted as *LOCATION* include named heavenly bodies, continents, countries, provinces, counties, cities, regions, districts, towns, villages, neighborhoods, airports, highways, street names, street addresses, oceans, seas, straits, bays, channels, sounds, rivers, islands, lakes, national parks, mountains, fictional or mythical locations, and monumental structures.

Input Text: *"Mustafa Kemal Bulvarı"* (*Mustafa Kemal Boulevard*)
Extraction: *<LOCATION> Mustafa Kemal Bulvarı </LOCATION>*

Input Text: *"Hükümet Caddesi"* (*Hukumet Street*)
Extraction: *<LOCATION> Hükümet Caddesi </LOCATION>*

Input Text: *"Yüksekova"* (*Yuksekova*)
Extraction: *<LOCATION> Yüksekova </LOCATION>*

Input Text: *"Musul"* (*Mosul*)
Extraction: *<LOCATION> Musul </LOCATION>*

Input Text: *"Londra"* (*London*)
Extraction: *<LOCATION> Londra </LOCATION>*

Input Text: *"E-5 Otoyolu"* (*E-5 Highway*)
Extraction: *<LOCATION> E-5 Otoyolu </LOCATION>*

Input Text: *"Erzincan-Sivas Demiryolu"* (*Erzincan-Sivas Railroad*)
Extraction: *<LOCATION> Erzincan-Sivas Demiryolu </LOCATION>*

Input Text: *"Amanos Dağı"* (*Amanos Mountain*)
Extraction: *<LOCATION> Amanos Dağı </LOCATION>*

Input Text: *"Uzungeçit Yaylası"* (*Uzungecit Plateau*)
Extraction: *<LOCATION> Uzungeçit Yaylası </LOCATION>*

Input Text: *"Cehennem Deresi"* (*Cehennem Creek*)
Extraction: *<LOCATION> Cehennem Deresi </LOCATION>*

Input Text: *"Sason Çayı"* (*Sason Stream*)
Extraction: *<LOCATION> Sason Çayı </LOCATION>*

Input Text: *"Atatürk Havalimanı"* (*Ataturk Airport*)
Extraction: *<LOCATION> Atatürk Havalimanı </LOCATION>*

Input Text: *"Fatma Girik Parkı"* (*Fatma Girik Park*)
Extraction: *<LOCATION> Fatma Girik Parkı </LOCATION>*

4.1.6 Temporal Expressions

- The *DATE* sub-type is a temporal unit of a full day or longer. Both absolute and relative dates are extracted as *DATE*.

Input Text: *"14 Haziran, 2004"* (*14 June, 2004*)
Extraction: *<DATE> 14 Haziran , 2004 </DATE>*

Input Text: *"21 ağustos"* (*21 August*)
Extraction: *<DATE> 21 ağustos </DATE>*

Input Text: *"20 ağustos - 20 eylül 2005"* (*20 August - 20 September 2005*)
Extraction: *<DATE> 20 ağustos - 20 eylül 2005 </DATE>*

Input Text: *"07 Mart 2006 Salı"* (*07 March 2006 Tuesday*)
Extraction: *<DATE> 07 Mart 2006 Salı </DATE>*

Input Text: *"dün"* (*yesterday*)
Extraction: *<DATE> dün </DATE>*

Input Text: *"bugün"* (*today*)
Extraction: *<DATE> bugün </DATE>*

Input Text: *"5 ay önce"* (*5 months ago*)
Extraction: *<DATE> 5 ay önce </DATE>*

- The *TIME* sub-type is defined as a temporal unit shorter than a full day, such as second, minute, or hour. Both absolute and relative times are extracted as *TIME*.

Input Text: *"bu sabah"* (*this morning*)
Extraction: *<TIME> bu sabah </TIME>*

Input Text: *"14:30"*
Extraction: *<TIME> 14:30 </TIME>*

Input Text: *"06.00"*
Extraction: *<TIME> 06.00 </TIME>*

- Temporal expressions are to be extracted as a single item. Contiguous subparts (month/day/year) are not to be separately extracted unless they are taggable expressions of two distinct temporal sub-types (date followed by time or time followed by date).

Input Text: *"28 Ekim, 2004 07:51:00 (TSİ)"* (*28 October, 2004 07:51:00 (Turkish Time Zone)*)
Extraction: *<DATE> 28 Ekim, 2004 </DATE> <TIME> 07:51:00 (TSİ) </TIME>*

- Compound ("marker-plus-unit") temporal expressions, and their lexicalized equivalents, should be extracted as single items. However, if a lexicalized "marker-plus-unit" modifies a contiguous time unit of a different sub-type, they should be tagged as two items.

Input Text: *"önceki gece"* (*the night before last night*)
Extraction: *<TIME> önceki gece </TIME>*

Input Text: *"dün sabah"* (*yesterday morning*)
Extraction: *<DATE> dün </DATE> <TIME> sabah </TIME>*

- Words or phrases modifying the expressions (such as "around" or "about") are not to be extracted.

Input Text: *"saat 10.00 sıralarında"* (*around 10 o'clock*)
Extraction: *<TIME> 10.00 </TIME>*

- Absolute time expressions combining numerals and time-unit designators or other subparts associated with a single temporal sub-type, are to be tagged as a single item.

Input Text: *"15:35:00 (TSİ)"* (*15:35:00 Turkish Time Zone)*)
Extraction: *<TIME> 15:35:00 (TSİ) </TIME>*

- When a temporal expression contains both relative and absolute elements, the entire expression is to be extracted.

Input Text: *"geçen nisan"* (*the last April*)
Extraction: *<DATE> geçen nisan </DATE>*

Input Text: *"geçtiğimiz yıl 6 ağustosta"* (*on August 6th last year*)
Extraction: *<DATE> geçtiğimiz yıl 6 ağustosta </DATE>*

- Indefinite or vague temporal expressions are not be extracted.

Input Text: *"son zamanlarda"* (*recently*)
Extraction: *None*

Input Text: *"şimdi"* (*now*)
Extraction: *None*

4.2 Generalization Features

Two important criteria that determine the efficacy and the success of an extractor are *(1)* the ability to recognize unseen named entities, and *(2)* the ability to precisely distinguish name entities that belong to a named entity class from the other named entity classes and non-entity names. Both criteria require accurate generalization of the known named entities. Generalizing means to recognize the parts susceptible of being changed in new names, and represent them with generic placeholders. In our study, we generalize named entities by using a set of features that are capable of describing various properties of the text. In addition to accurate generalization, the use of generalization features will help overcome the data sparseness problem that occurs because of the diversity of the language constructs and the insufficiency of the input data. When we consider all possible language constructs, it is not possible to observe most of the sequences during the training of the language model.

The features used in our study can be grouped into the following four categories:

- *Lexical Features*: IE deals with text documents which can be seen as contiguous series of tokens. As basic constituents of the text, the tokens themselves are used for IE as well as the features associated with them. Gazetteer information (e.g. list of person names, list of city names) provided to the system can be mapped to the tokens and utilized for generalization purposes. We used a two level gazetteer hierarchy in our study to achieve accuracy in generalization. The first level in our hierarchy corresponds to each named entity class (e.g. *Person*, *Location*) and provides a higher level of generalization. The second level details the gazetteer categorization (e.g. *Location.Country*, *Location.City*) and provides more specific classification. The complete list of the used gazetteer lists is provided in Appendix D.

- *Morphological Features*: We effectively used the morphological features of the tokens not only for addressing the challenges arising from the agglutinative nature of Turkish, but also for the clues they offer towards better generalization.

- *Contextual Features*: The information captured in the surrounding text of the named entities is used by the system to learn and represent the patterns and regularities in the target named entity boundaries which exist in the training dataset.

- *Orthographic Features*: These features express various orthographic aspects of the tokens. We selected four primitive features, a combination of which can yield more complicated patterns: Capitalization (*Lower*, *Upper*, *Proper*, *Unclassified*), Length Class (*Short*, *Middle*, *Long*), Length (the length of the token), and Type Class (*Alpha*, *Numeric*, *Alphanumeric*, *Punctuation*, *Unclassified*).

4.3 Rule Representation

The ability to recognize the regularities among the target named entities and to capture the learnt regularities/patterns in the rules requires a powerful rule representation. Moreover, the coverage of the learnt rules which also related rule representation affects the performance of the rule learning systems. While over-specific rules may cause low recall, over-general rules cause low precision. An expressive rule language that can handle the mentioned generalization features in a flexible manner and provide the adequate level of granularity for rule coverage is necessary to achieve good NER performance.

A NER rule defined in our system consists of four parts. The first part simply addresses the NE class which is the target of this rule. The last three parts contain the pattern segments: (1) The *PRE-FILLER (PRE)* segment tries to sense and match the text immediately preceding the target NE, (2) The *FILLER (FILL)* segment tries to sense and match the target NE, and (3) The *POST-FILLER (POST)* segment tries to sense and match the text immediately following the target NE. The pattern segments in an extraction rule can be seen as a sequence of pattern elements whose type can be either *Similarity (SIM)* or *Optional (OPT)*. A type *SIM* pattern element matches exactly one token from the document that meets the element's constraints. On the other hand, a type *OPT* pattern element can match either a token that meets the element's constraints or none. Being tailored parallel to the features mentioned in the previous section, our rule representation uses pattern elements containing several fields to provide the expressiveness we need for accurate generalization.

In order to make the rule representation concept clearer, two example rules are given in Figure 4.1. In our rule syntax, rule parts are separated by colons. As the first part of the first rule indicates, the rule captures the pattern information belonging to

➡ [PERSON :

SIM<valisi; vali+Noun+A3sg+P3sg+Nom; {Person.Title}; {Person}; Proper; Middle; 6; Alpha> :

SIM< ; *+Noun+?(Prop)+A3sg+Pnon+Nom; {Person.First_Name}; {Person}; Proper; ; ; Alpha> OPT< ; *+Noun+?(Prop)+A3sg+Pnon+Nom; {Person.First_Name}; {Person}; Proper; ; ; Alpha> SIM< ; *+?(Noun)+*+*+*+*; {Person.Last_Name}; {Person}; Proper; ; ; Alpha> :

SIM<,; ,+Punc; {};{}; Unclass; Short; 1; Punc>]

➡ [DATE :

NULL :

SIM< ; *+Num+Card; {Date.Day_Number, Time.Minute, Number.Number}; {Date, Time, Number}; Unclass; Short; 2; Number> SIM< ; *+Noun+A3sg+Pnon+Nom; {Date.Month}; {Date}; ; ; ; Alpha> OPT<,; ,+Punc; {}; {}; Unclass; Short; 1; Punc> SIM<; *+Num+Card; {Date.Year}; {Date}; Unclass; Short; 4; Number> :

SIM< ; *+Noun+A3sg+P3sg+*; {Date.Post_Phrase}; {Date}; Lower; ; ; Alpha>]

Figure 4.1: Example NER rules

PERSON NE class. The next part, the *PRE* segment, in the first example contains only one pattern element, a *SIM* element, which has eight fields separated by semicolons: (1) token, (2) morphological tag, (3) low-level gazetteer set, (4) high-level gazetteer set, (5) case tag, (6) length class, (7) token length, and (8) type class. Each field in the pattern element represents a constraint. In order a pattern element to match a token in the text, that token must satisfy all of the constraints imposed by that pattern element (conjunction of the constraints).

The first constraint, token field, is a non-case sensitive atomic field that matches only the exact text values, if it is set to a value. The morphological tag stored in the second field is a simple regular expression that provides a concise and flexible means for matching tokens according to their morphological characteristics. Each morpheme in the morphological tag is separated by a plus sign. Special characters are used to express the specific variations in the regular expressions. The star (*) character matches a single morpheme with any type. The question mark (*?*) character with the combination of parentheses indicates that there is zero or one of a single morpheme with the morpheme

type given in the parentheses. When it is used with the star character (i.e. *"?(*)"*), it indicates that there is zero or one of a single morpheme with any type. The third and the fourth field contain gazetteer set constraints. To be matched, a token must be in the coverage of the gazetteer sets (i.e. a token must not be included any gazetteer list which is not in the sets). Since it is practically impossible to have complete gazetteer lists that cover all available names, our matching algorithm does not refuse the tokens which are not listed in any of our gazetteer lists in order to enable the system to tolerate the potential faults that can be caused by missing names in the gazetteer lists. The remaining atomic fields contain the constraints regarding to the orthographic features described in the preceding section. Returning to the first example given in Figure 4.1, the *PRE* segment of the rule indicates that the named entity phrase to be matched by this rule must be preceded by the token *Valisi (Governor (of))*.

- Example Rule 1 (*Person*):
  - ➨ ...Adana Valisi Cahit Kıraç, Türk-Amerikan Derneği binasındaki patlamanın konulan bombadan...
  - ➨ ...İstanbul Valisi Muammer Güler, Çapa'da İETT otobüsünde...
  - ➨ ...inceleme yapan Ağrı Valisi Halil İbrahim Akpınar, yangının terör...
  - ➨ ...Bursa Valisi Oğuz Kağan Köksal, Büyükşehir Belediye Başkanı Hikmet...

- Example Rule 2 (*Date*):
  - ➨ ...Van'da 31 ekim 2005 tarihinde Erek Polis Karakolu'na...
  - ➨ ...20 Mayıs 2003 tarihinde Ankara Kızılay'da bir kafede...
  - ➨ ...göre, 26 Eylül 2000 günü akşam saatlerinde...

Figure 4.2: Text excerpts containing named entities that match the example rules given in Figure 4.1

The *FILL* segment states that the named entity phrase to be matched by this rule must start with a token which is a name in nominative form, not listed in any gazetteer list other than *Person.First_Name* gazetteer list, in proper case, in any length and containing only alpha characters. The rule also requires that the last token of the person phrase must be either in *Person.Last_Name* gazetteer list or none, in proper case, in any length and containing only alpha characters. Optionally, another token possessing the same characteristics as the first token can occur between the first and the last token. Finally, the *POST* segment asserts that the named entity phrase to be matched by the rule must be followed by a comma. The rule given in the second

example describes a pattern belongs to *DATE NE* class in a similar fashion. Note that the *PRE* segment is set to *NULL* value, which means the rule does not impose any constraints on the tokens that can occur before the target named entity phrase. Some text excerpts containing named entities that match the given example rules are shown in Figure 4.2.

4.4 Automatic Rule Learning

The ability of rule learning and subsequent generalization is one of the critical functions in the system. Our automatic rule learning and generalization method is based on the concept of specific generalization of strings [20]. We applied the concept to generalize both the patterns and the features in different levels; and employed a modified version of the coverage algorithm presented in the study for inducing the rules. In order to generalize two strings, a unique match sequence of those strings is obtained, and the differences in the unique match sequence are replaced by variables to get a generalized form of the strings. Referring the reader to the previous chapter for the details of the concept, we will focus more on how we adapted the concept to automatic rule learning for NER.

Prior to learning NER rules from the examples in the training text, the input text is segmented in sentences and tokenized. We followed the standard tokenization method which uses white-space and punctuation characters as delimiters except that we removed the apostrophe symbol (') from our delimiter set since morphemes coming after the named entities are considered as part of the name in our study. The next step is to assign feature values to every token in the text. First, possible morphological parses of each observed token are found, and the most appropriate one among the found parses is selected through morphological disambiguation process [24, 61]. Upon labeling the observed tokens with the appropriate morphological tags, the system continues with gazetteer list search. An important point to highlight is that the stem of the token is looked up in the gazetteer lists to minimize the effect of morphological complexity. Finally, each token is labeled with their corresponding orthographic feature values before starting the learning. Thus each token is represented by its eight features, and NER rules are learnt from these representations of tokens.

Subsequent to preprocessing of the training data, the learner starts generating simple patterns from the training examples. A simple pattern is generated using the preceding token, the following token, and the tokens of a named entity. The generated simple patterns are kept in the rule representation stated in the previous section. The generalization function $GEN(R_1, R_2)$ takes two examples in the rule representation and returns a generalized rule that covers the both examples by relaxing the corresponding constraints specified by the feature fields in the pattern elements. The generalization of the atomic fields (token, capitalization, length class, length, type class) is straightforward. If the values in the corresponding fields of the examples are the same, this value is assigned to the same field of the generalized pattern element; otherwise, the field is left empty, which indicates that no constraint is defined for that field. Generalization of morphological tag field is based on the concept of specific generalization of strings [20]. Processing similarities and differences between the morphological tag fields of the examples, and replacing the differences with either a type ANY (*) variable or a type OPTIONAL (?) variable, or combination of these two, the function obtains a simple regular expression that can represent the both examples. For gazetteer set fields, the generalization operation returns the union of two input gazetteer sets. However, if one of the gazetteer sets is empty, the generalization operation returns an empty set. Empty gazetteer sets impose no constraint on the patterns.

Because the examples can differ in the number of tokens, we applied a method that calculates a simple similarity score for each possible way for matching the tokens and selects the match with the maximum similarity score. The similarity score for a match is the sum of the similarity scores of each pattern element, which is the aggregated sum of the similarity scores of each feature field. Each field's contribution to the similarity score vary according to its discrimination power. For instance, the orthographic type class is apparently less powerful than gazetteer list sets in discriminating NEs.

In order to illustrate the rule generalization concept, an example rule generation is given in Figure 4.2. In the example, a generalized rule is learnt from two person name instances located in the following text excerpts: "...Elazığ Valisi Kadir Koçdemir'in geçtiği...", "...Van Valisi Hikmet Tan, konvoyuna...". By performing several generalization operations over the different pattern elements, the learner obtains the generalized rule shown in Figure 4.2. In the example, the generalized rule asserts the following constraints: (1) the named entity phrase to be matched by this rule must be preceded by the token Valisi, (2) the named entity phrase to be matched by this rule

- Seed Instances (*Person*):

  "...Valisi Kadir Koçdemir'in geçtiği...", "...Valisi Hikmet Tan,..."

- Preprocessing:

  ➨ <valisi; vali+Noun+A3sg+P3sg+Nom; {Person.Title}; {Person}; Proper; Middle; 6; Alpha>

  <kadir; kadir+Noun+Prop+A3sg+Pnon+Nom; {Person.First_Name, Person.Last_Name}; {Person}; Proper; Middle; 5; Alpha>
  <koçdemir'in; koçdemir+Noun+Prop+A3sg+Pnon+Nom; {Person.Last_Name}; {Person}; Proper; Long; 11; Alpha>

  <geçtiği; geç+Verb+Posˆ DB+Adj+PastPartˆ DB+Noun+Zero+A3sg+P3sg+Nom; { }; { }; Lower; Middle; 7; Alpha>

  ➨ <valisi; vali+Noun+A3sg+P3sg+Nom; {Person.Title}; {Person}; Proper; Middle; 6; Alpha>

  <hikmet; hikmet+Noun+A3sg+Pnon+Nom; {Person.First_Name}; {Person}; Proper; Middle; 6; Alpha>
  <tan; tan+Noun+A3sg+Pnon+Nom; {Person.First_Name, Person.Last_Name}; {Person}; Proper; Short; 3; Alpha>

  <,; ,+Punc; { }; { }; Unclass; Short; 1; Punc>

- Simple Pattern#1

  [PERSON:

  SIM<valisi; vali+Noun+A3sg+P3sg+Nom; {Person.Title}; {Person}; Proper; Middle; 6; Alpha> :

  SIM<kadir; kadir+Noun+Prop+A3sg+Pnon+Nom; {Person.First_Name, Person.Last_Name}; {Person}; Proper; Middle; 5; Alpha> SIM<koçdemir'in; koçdemir+Noun+Prop+A3sg+Pnon+Nom; {Person.Last_Name}; {Person}; Proper; Long; 11; Alpha>:

  SIM <geçtiği; geç+Verb+Posˆ DB+Adj+PastPartˆ DB+Noun+Zero+A3sg+P3sg+Nom; { }; { }; Lower; Middle; 7; Alpha>]

- Simple Pattern#2

  [PERSON:

  SIM<valisi; vali+Noun+A3sg+P3sg+Nom; {Person.Title}; {Person}; Proper; Middle; 6; Alpha>:

  SIM<hikmet; hikmet+Noun+A3sg+Pnon+Nom; {Person.First_Name}; {Person}; Proper; Middle; 6; Alpha>

  SIM<tan; tan+Noun+A3sg+Pnon+Nom; {Person.First_Name, Person.Last_Name}; {Person}; Proper; Short; 3; Alpha>

  SIM<,; ,+Punc; { }; { }; Unclass; Short; 1; Punc>]

- Generalized Rule:

  [PERSON :

  SIM<valisi; vali+Noun+A3sg+P3sg+Nom; {Person.Title}; {Person}; Proper; Middle; 6; Alpha> :

  SIM< ; *+Noun+?(Prop)+A3sg+Pnon+Nom; {Person.First_Name, Person.Last_Name}; {Person}; Proper; Middle; ; Alpha>

  SIM< ; *+Noun+?(Prop)+A3sg+Pnon+*; {Person.First_Name, Person.Last_Name}; {Person}; Proper; ; ; Alpha> :

  SIM< ;*+?(Verb)+?(Pos)+?(^DB)+?(Adj)+?(PastPart)+?(^DB)+?(Noun)+?(Zero)+?(A3sg)+?(P3sg)+*; { }; { }; ; ; ; ; >]

- Recognizable NEs:

  “Adana Valisi Cahit Kıraç,”, “Tunceli Valisi Mustafa Erkal yaptığı”, “Çankırı Valisi Ayhan Çevik’in bulunduğu”, “İstanbul Valisi Muammer Güler,”

Figure 4.2: An example NER rule generation

must start with a token which is a name in nominative form, listed in Person.First_Name and/or Person.Last_Name gazetteer lists, in proper case, 5-8 characters in length and containing only alpha characters, (3) the last token of the named entity must be in Person.First_Name and/or Person.Last_Name gazetteer lists, in proper case, in any length and containing only alpha characters, and (4) the token comes after the named entity must match the morphological tag specified by the POST segment of the rule. Note that some person names recognizable by the generalized rule are also given in the example. As evident from the given list, exploiting morphological features increases the recognizability of the NEs.

Our coverage algorithm that finds *RULES*, the set of generalized rules, is given in Figure 4.3. Initially, *RULES* is set to an empty set (Figure 4.3, Line 1). The algorithm then generates the rule R for all positive examples available in the training dataset and adds R into *RULES* if it is not already in the set (Figure 4.3, Lines 2-4). Afterwards, the algorithm iteratively generalizes the rules available in *RULES* (Figure 4.3, Line 5-19). In each iteration, for each rule R_1 in *RULES*, possible generalized rules are found and kept in a temporary rule-set $RULES_{temp}$ (Figure 4.3, Lines 9-12). Then, the rules in $RULES_{temp}$ are sorted in descending order of the similarity factor, a score that is directly proportional to the similarity of the rules used to generate a generalized rule (Figure 4.3, Line 13). The sort process is performed to ensure that the rules with high similarity factors are added into *RULES* in the next step. Subsequently, until k number of generalized rules are added into *RULES* or every rule in $RULES_{temp}$ are validated, the rules in $RULES_{temp}$ are validated on the training dataset in order to give their confidence factors; and the rules with confidence factors above a certain threshold value are added into *RULES*, while the rules from which the generalized rule generated are dropped (Figure 4.3, Lines 14-19). The confidence factor of a rule is calculated as the percentage of correctly extracted names as a result of applying that rule to the training dataset. During the confidence factor calculation, the algorithm also collects the rule exceptions and builds the rule exception list for each rule, which we will discuss in the next section. This iterative loop continues until no more generalized rule with confidence factor above the threshold value can be added into *RULES*. After sorting the *RULES* in ascending order of the coverage -number of positive examples covered- (Figure 4.3, Line 10), the algorithm eliminates the redundancy in *RULES*. If all positives examples covered by a rule are also covered by some other rules in the rule-set, that rule is deleted from the set (Figure 4.3, Lines 11-15). The reasoning behind sorting rules in ascending order of their coverage is our preference of general

rules to specific rules.

(1) $RULES \leftarrow \varnothing$
(2) ***FOR*** each positive example e in the example space E:
(3) $R \leftarrow$ e
(4) ***IF*** $R \notin RULES$ ***ADD*** R into $RULES$
(5) $hasMoreGeneralization \leftarrow TRUE$
(6) ***WHILE*** $(hasMoreGeneralization = TRUE)$
(7) $hasMoreGeneralization \leftarrow FALSE$
(8) ***FOR*** each rule $R_1 \in RULES$:
(9) $RULES_{temp} \leftarrow \varnothing$
(10) ***FOR*** each rule $R_2 \in RULES$ (where $R_1 \neq R_2$):
(11) $R \leftarrow GEN(R_1, R_2)$
(12) ***IF*** $R \notin RULES_{temp}$ ***ADD*** R into $RULES_{temp}$
(13) ***SORT*** $RULES_{temp}$ in descending order of the similarity factor
(14) ***UNTIL*** k rules added into $RULES$ ***or*** every rule in $RULES_{temp}$ validated:
(15) ***TEST*** every rule R in $RULES_{temp}$ on the training dataset and ***CALCULATE*** CF_R (the confidence factor of R)
(16) ***IF*** $CF_R \geq T$ (confidence factor threshold) ***and*** $R \notin RULES$
(17) ***ADD*** R into $RULES$
(18) ***DROP*** rules R_1 and R_2, from which R generalized, from $RULES$
(19) $hasMoreGeneralization \leftarrow TRUE$
(20) ***SORT*** $RULES$ in ascending order of the coverage
(21) ***FOR*** each rule $R \in RULES$:
(22) ***IF*** there exists another rule that covers E_R (the positive examples covered by R)
(23) ***DROP*** R from $RULES$
(24) ***IF*** every example in E_R is also covered by another rule in $RULES$
(25) ***DROP*** R from $RULES$

Figure 4.3: The rule generalization algorithm

There are two parameters determined by the user: the confidence factor threshold (T), and the number of generalized rules to generate for each rule in each cycle (k). The first parameter controls the trade-off between selectivity and sensitivity. By setting a higher T value, it is possible to increase the system's ability to precisely distinguish name entities that belong to a named entity class from the other named entity classes and non-entity names; however this can result in a decrease in the system's ability to recognize unseen named entities. The second parameter controls the learning time of the system. The confidence factor of a rule is found by validating that rule on the training dataset, which requires a computational time. By limiting the number of generalized rules to generate for each rule in each cycle, the algorithm provides control over the total computational time spent for confidence factor calculation.

4.5 Rule Refinement

In order to make full use of the information available in training data and improve the algorithm's extraction performance by further rule refinement, each rule in the rule-set is associated with a set of exceptions. The problem is that of efficient utilization of the negative examples (i.e. non-NEs or NEs of different classes) in the training data, though they are used in confidence factor calculation (Figure 4.3, Line 15). Unless it is a 100% confident rule, a rule in the final rule-set may cover some negative instances in the training data. This leads to recognition of an incorrect NE during the test, even if that name is marked as a non-NE or a NE of a different class in the training data. This issue is solved by associating each rule in the final rule-set with a set of exceptions. During confidence factor calculation, every negative instance recognized by the candidate rule is put into that rule's exception set. If any of the names in a rule's exception set are recognized by that rule during the test, the recognized names are just ignored and not extracted.

4.6 Testing & Post-Processing

Subsequent to the generation of the rule-set and the completion of the training phase, the test phase starts. Starting from each token in the text, the system applies the learnt rules to the test data. If a rule matches a phrase between two sentence boundaries and the matched phrase is not in that rule's exception set, the matched phrase is put into a candidate list. In case of any conflict (i.e. overlapping phrases), the longest candidate is selected during the post-processing step, which comes after testing.

Chapter 5

Entity Relation Detection

5.1 Task Definition

Entity Relation Detection refers to identifying relationships between entities mentioned in text documents. In our study, we restrict the scope of the problem to sentence level. That is, our relationship detection system identifies and extracts relationships at the sentence level.

Although we generally followed the existing MUC-7 information task definition [18] as a guideline, the task definiton is tailored to the scope and the objectives of our study. However, these adaptations are mostly domain related. For instance, relation categories in our study are different from those defined in [18]. The following subsections describe details of our task definiton.

5.1.1 Scope & General Guidelines

The task is restricted to the following three relation types possible between ORGANIZATION, PERSON, and LOCATION named entity categories:

- *LOCATED_IN*: indicates an *ORGANIZATION* or a *LOCATION* is *located in* another *LOCATION*.

- *AFFILIATED_WITH*: shows a *PERSON* is *affiliated with* an *ORGANIZATION*.
- *ATTACKED_BY*: indicates that a *PERSON* or an *ORGANIZATION* is *attacked by* another *PERSON* or another *ORGANIZATION*.

The target relations are required to be between two entities in the same sentence. Moreover, the extracted information must be locatable or linked to something in the text. Our scope does not cover the use of world knowledge or the format of the article in inferencing during information extraction.

Some entities may be in more than one relationship. Our task definition requires the extraction of all relations available in the text.

5.1.2 LOCATED_IN Relations

- Relationship between two locations: A *LOCATION* is located in another *LOCATION*.

Input Text: *"Bingöl'ün Genç ilçesinde, PKK'lı teröristler tarafından döşenen mayın, askeri aracın geçişi sırasında patladı."* (*The mine planted by PKK terrorists exploded during military vehicle's pass in Bingol's Genc district.*)
Extracted Relation: *<LOCATED_IN Entity="Genç" Ref_Entity="Bingöl'ün" />*

- Relationship between an organization and a location: An *ORGANIZATION* is located in a *LOCATION*.

Input Text: *"Mardin'de Yenişehir Polis Karakoluna teröristlerce saldırıldı."* (*Yenisehir Police Station was attacked by terrorists in Mardin.*)
Extracted Relation: *<LOCATED_IN Entity="Yenişehir Polis Karakoluna" Ref_Entity= "Mardin'de" />*

- Transitive case: An *ORGANIZATION* is located in a *LOCATION* which is located in another *LOCATION*. Hence, the *ORGANIZATION* is also located in the second *LOCATION*.

Input Text: *"Ağrı'nın Doğubeyazıt ilçesindeki Uluyol Polis Merkezi'ne saldırı düzenlendi."* (*In Agri's Dogubeyazit district, an attack against Uluyol Police Center was organized.*)

Extracted Relation: *<LOCATED_IN Entity="Doğubeyazıt "*
Ref_Entity="Ağrı'nın" />

Extracted Relation: *<LOCATED_IN Entity="Uluyol Polis Merkezi'ne"*
Ref_Entity="Doğubeyazıt" />

Extracted Relation: *<LOCATED_IN Entity="Uluyol Polis Merkezi'ne"*
Ref_Entity="Ağrı'nın" />

5.1.3 AFFILIATED_WITH Relations

- Relationship between a person and an organization: A *PERSON* is affiliated with an *ORGANIZATION*.

Input Text: *"Fail DHKP-C terör örgütü üyesi Gültekin Koç olarak belirlendi."* (*The perpetrator was identified as DHKP-C terrorist organization member Gultekin Koc.*)

Extracted Relation: *<AFFILIATED_WITH Entity="Gültekin Koç"*
Ref_Entity="DHKP-C" />

- Relationships between multiple persons and an organization: Multiple *PERSON*s are affiliated with the same *ORGANIZATION*.

Input Text: *"Patlamada, TCDD çalışanları Celal Korkmaz, Özcan Türker, ve Mehmet Şimşek hayatını kaybetti."* (*In the explosion, TCDD employees Celal Korkmaz, Ozcan Turker, and Mehmet Simsek* lost their lives.)

Extracted Relation: *<AFFILIATED_WITH Entity="Celal Korkmaz"*
Ref_Entity="TCDD" />

Extracted Relation: *<AFFILIATED_WITH Entity="Özcan Türker"*
Ref_Entity="TCDD" />

Extracted Relation: *<AFFILIATED_WITH Entity="Mehmet Şimşek"*
Ref_Entity="TCDD" />

5.1.4 ATTACKED_BY Relations

- Relationship between a person and an organization: A *PERSON* is attacked by an *ORGANIZATION*.

Input Text: *"Vali Çevik'e yapılan saldırıyı terör örgütü TKP-ML/TİKKO üstlendi."* (*The terrorist organization TKP-ML/TIKKO claimed the attack against the governor Cevik.*)

Extracted Relation: *<ATTACKED_BY Entity="Çevik'e" Ref_Entity="TKP-ML/TİKKO" />*

- Relationship between two organizations: An *ORGANIZATION* is attacked by another *ORGANIZATION*.

Input Text: *"PKK tarafından Erenkaya Köyü Jandarma Karakoluna dün gece saldırı düzenlendi."* (*An attack was organized against Erenkaya Village Gendarmerie Station by PKK last night.*)

Extracted Relation: *<ATTACKED_BY Entity="Erenkaya Köyü Jandarma Karakoluna" Ref_Entity="PKK" />*

- Relationship between an organization and a person: An *ORGANIZATION* is attacked by a *PERSON*.

Input Text: *"Danıştay'a yapılan saldırının failinin Alparslan Aslan olduğu açıklandı."* (*It was announced that the perpetrator of the attack against the Council of State is Alparslan Aslan.*)

Extracted Relation: *<ATTACKED_BY Entity="Danıştay'a" Ref_Entity="Alparslan Aslan" />*

- Relationship between two persons: A *PERSON* is attacked by another *PERSON*.

Input Text: *"Alparslan Aslan tarafından yapılan saldırı sonucunda Mustafa Yücel Özbilgin hayatını kaybetti."* (*Mustafa Yucel Ozbilgin lost his life as a result of the attack carried out by Alparslan Aslan.*)

Extracted Relation: *<ATTACKED_BY Entity="Mustafa Yücel Özbilgin" Ref_Entity="Alparslan Aslan" />*

5.2 Rule Representation

The rule representation for capturing the regularities among the relations is similar to the representation we utilized for NER rules. The major distinctions are (1) the token versus phrase level processing units and (2) the phrase versus sentence level pattern focus. Since named entities are phrases formed by a number of consecutive tokens, tokens in texts are the main processing units for NER. Moreover, the NER patterns focus on the phrases in the sentences. For ERD task, named entities are treated as a single processing unit and the patterns focus on the whole sentences.

A relation extraction rule consists of four parts. The first part is the target relation category of this rule. The second and the third parts show the referencing and the referenced entity respectively. The last part contains the pattern segment. The pattern segment consists of a number of pattern elements whose type can be (1) *Referencing Entity (ENT)*, (2) *Referenced Entity (REF_ENT)*, (3) *Named Entity/Temporal Expression (ORGANIZATION, PERSON, LOCATION, DATE, TIME)*, (4) *Similarity (SIM)*, and (5) *Skip (*)*. The *ENT* pattern element is the placeholder for the referencing entity. Similarly, the *REF_ENT* is the placeholder for the referenced entity. A *Named Entity or Temporal Expression* can match any named entity or temporal expression of its type. The *SIM* pattern element matches exactly one token from the document that meets the its constraints. The *Skip (*)* element is the most flexible pattern element which can match any phrase with any number of tokens, including NEs.

Two example rules are given in Figure 5.1 to describe the rule representation. The first rule captures the pattern information belonging to a *LOCATED_IN* relation between two *LOCATION*s. The pattern segment, in the first example contains eight pattern elements. It starts with a *Skip (*)* element followed by a *LOCATION* element. The *Skip (*)* element means to skip any number of processing units (e.g. tokens, named entity phrases) until the next occurence of the following term in the pattern. In this case, the pattern skips until it finds a *LOCATION* element. The third element *REF_ENT* is the placeholder for the referenced location. This element is followed by two *SIM* elements. *SIM* elements used in NER rule patterns and ERD rule patterns have exactly the same structure and specify the same constraints to match tokens. The next element *ENT* is the placeholder for the referencing location. The pattern continues with a *Skip (*)* element. The last element in the pattern is another *SIM*. As shown in the examples, *ENT*, *REF_ENT* and *Named Entity/Temporal Expression*

elements have a similar structure and contain a entity type field and a morphological tag field. The morphological tag field captures the morphological characteristics of the last tokens of the entity phrases.

```
➨ [ LOCATED_IN : LOCATION : LOCATION :
*
LOCATION<*+Noun+Prop+A3sg+*+*>
REF_ENT<LOCATION; *+?(Noun)+?(*)+?(*)+?(*)+?(*)+?(*)>
SIM<; ilçe+Noun+?(Prop)+A3sg+P3sg+Dat; {Location.Post_Phrase}; {Location}; ;
; ; Alpha>
SIM<bağlı; bağ+Noun+A3sg+Pnon+Nom+^DB+Adj+With; {Location.Pre_Phrase,
Organization.First_Phrase, Organization.In_Phrase, Organization.Post_Phrase};
{Location, Organization}; Lower; Middle; 5; Alpha>
ENT<LOCATION; *+Noun+?(*)+?(*)+?(*)+?(*)+?(*)+?(*)+?(*)+?(*)>
*
SIM<.;.+Punc; {Person.Short_Title, Person.In_Phrase}; {Person}; Unclass; Short; 1;
Punc> ]

➨ [ ATTACKED_BY : ORGANIZATION : ORGANIZATION :
*
ENT<ORGANIZATION; *+Noun+?(*)+*+?(*)+Dat>
*
SIM<terör; terör+Noun+A3sg+Pnon+Nom; {}; {}; Lower; Middle; 5; Alpha>
SIM<örgütü; örgüt+Noun+A3sg+P3sg+Nom; {Organization.Last_Phrase};
{Organization}; Lower; Middle; 6; Alpha>
REF_ENT<ORGANIZATION; *+Noun+*+*+*+*>
*
SIM<saldırı; saldırı+Noun+A3sg+Pnon+Nom; {}; {}; Lower; Middle; 7; Alpha>
*
SIM<.; .+Punc; {Person.Short_Title, Person.In_Phrase}; {Person}; Unclass; Short; 1;
Punc> ]
```

Figure 5.1: Example ERD rules

The second rule captures the pattern information belonging to a *ATTACKED_BY* relation between two *ORGANIZATION*s. In this example, the pattern segment contains ten pattern elements. The rule implies (1) skip until reaching an *ORGANIZATION*; (2) extract the found *ORGANIZATION* as the referencing entity; (3) skip until finding the string *"terör"* followed by the string *"örgütü"* and match the found strings; (4) if the next element is an *ORGANIZATION*, extract it as the referenced entity; (5) skip until finding the string *"saldırı"* and match the found string; (6) skip until finding the character *"."* and match it. A number of sentences containing relations that match the given example rules are shown in Figure 5.2.

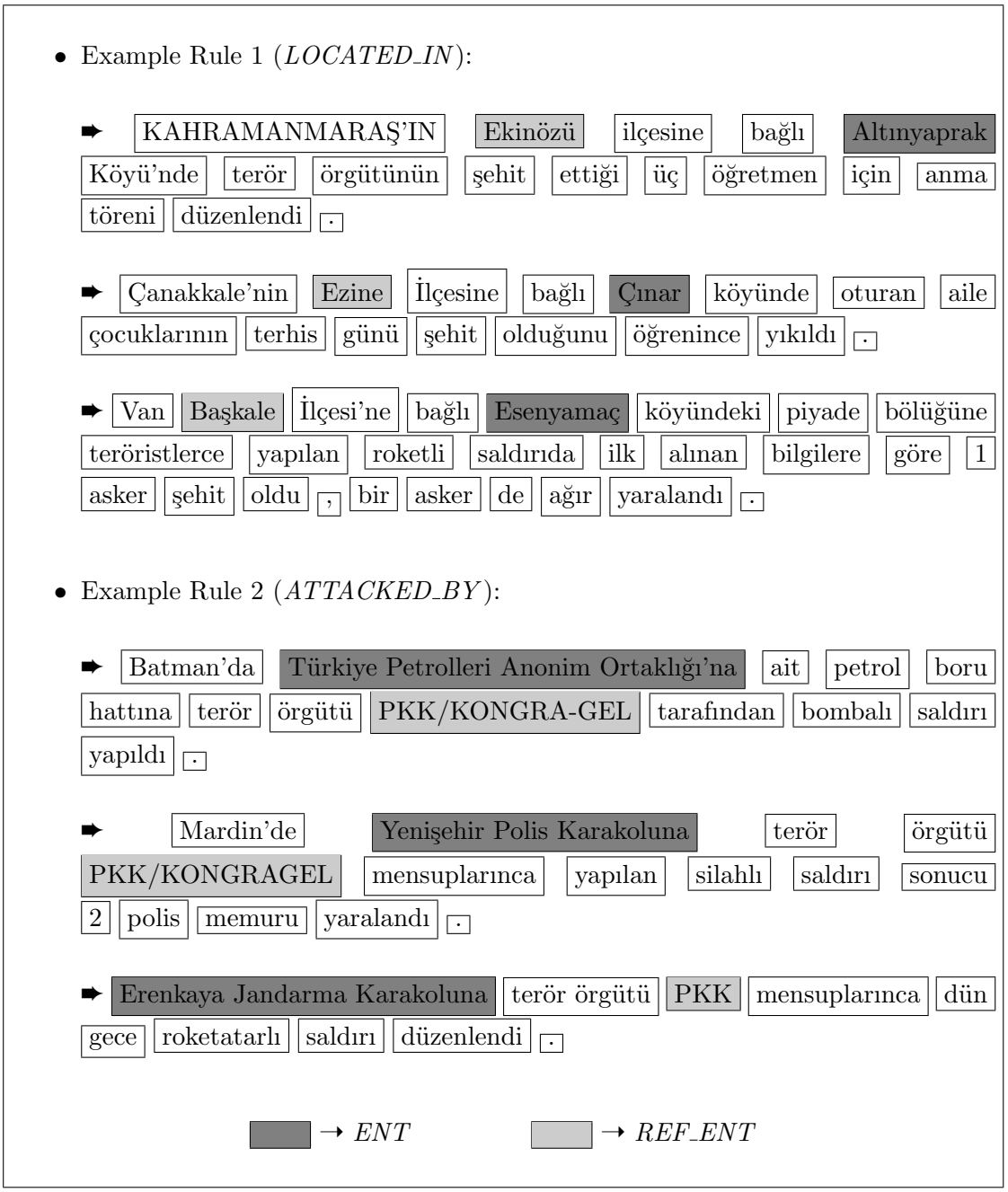


Figure 5.2: Sentences containing relations that match the example rules given in Figure 5.1

5.3 Automatic Rule Learning

Our automatic rule learning and generalization method for ERD is also based on the concept of specific generalization of strings [20]. The concept is further adapted to generalize the relation patterns. The same coverage algorithm (Figure 4.3) used for NER is also used for ERD task.

Prior to learning ERD rules from the examples in the training text, the input text is tokenized, segmented in sentences and the entity names in the text are identified. Then, the learner starts generating simple relation patterns from the training examples. The generated simple patterns are kept in the rule representation stated in the previous section. The output of the generalization operation for two rules is a generalized rule that covers the both examples. The generalization operation is performed by relaxing the corresponding constraints specified by the pattern elements in the rules. The generalization of the pattern elements of different types (e.g. a *Named Entity* type and a *SIM* type) results in a *Skip (*)* element. On the other hand, the generalization of the pattern elements of the same type outputs the pattern type of the input pattern elements. A generalization with a *Skip (*)* element always results in a *Skip (*)* element. For *SIM* elements, the atomic fields (token, capitalization, length class, length, type class) are generalized in the same way as generalized in NER. The same situation is also valid for the morphological tag and gazetteer set fields.

In order to illustrate the rule generalization concept, an example rule generation is given in Figure 5.2. In the example, a generalized rule is learnt from two *LOCATED_IN* relations. First, the simple patterns representing the input relations are generated. Since no generalization operation is performed over the simple patterns yet, they don't include any *Skip (*)* element. By performing several generalization operations over the simple patterns, the learner obtains the generalized rule shown in Figure 5.2. The generalized rule asserts that a *LOCATION* immediately preceded by another *LOCATION* is *LOCATED_IN* the preceeding *LOCATION*. The new relations recognizable by the generated rule is also given in the example.

5.4 Testing & Post-Processing

The testing and the post-processing steps are performed in the same manner as done in NER. For each sentence in the text, the system applies the learnt rules to the test data. During testing, the found relations are put into a candidate list. In case of any conflict (i.e. two different relations between the same entities), the candidate added by the rule with the highest confidence factor is selected during post-processing.

- Seed Instances (*LOCATED_IN*):
  - ➨ Gümüşhane'nin | Şiran | ilçesinde | teröristler | askeri | araca | ateş | açtı | .
  - ➨ İstanbul | Bağcılar'da | polis | ekibine | silahlı | saldırı | düzenlendi | .
- Simple Pattern#1

  [LOCATED_IN : LOCATION : LOCATION :
  REF_ENT<LOCATION; gümüşhane+Noun+Prop+A3sg+Pnon+Gen>
  ENT<LOCATION; şiran+Noun+Prop+A3sg+Pnon+Nom>
  SIM<ilçesinde; ilçe+Noun+A3sg+P3sg+Loc; {Location.Post_Phrase}; {Location}; Lower; Long; 9; 1>
  SIM<teröristler; terörist+Noun+A3pl+Pnon+Nom; {}; {}; Lower; Long; 11; Alpha>
  SIM<askeri; askeri+Adj; {Location.Vilage, Organization.First_Phrase, Organization.In_Phrase}; Location, Organization; Lower; Middle; 6; Alpha>
  SIM<araca; araç+Noun+A3sg+Pnon+Dat; {Person.Last_Name, Location.District}; {Per, Location}; Lower; Middle; 5; Alpha>
  SIM<ateş; ateş+Noun+A3sg+Pnon+Nom; {Person.First_Name, Person.Last_Name, Person.Title}; {Person}; Lower; Short; 4; Alpha>
  SIM<açtı; aç+Verb+Pos+Past+A3sg; {Person.First_Name}; {Person}; Lower; Short; 4; Alpha>
  SIM<.; .+Punc; {Person.Short_Title, Person.In_Phrase}; {Person}; Unclass; Short; 1; Punc>]

- Simple Pattern#2

  [LOCATED_IN : LOCATION : LOCATION :
  REF_ENT<LOCATION; istanbul+Noun+Prop+A3sg+Pnon+Nom>
  ENT<LOCATION; bağcılar+Noun+Prop+A3sg+Pnon+Loc>
  SIM<polis; polis+Noun+A3sg+Pnon+Nom; {Person.Title, Organization.First_Phrase, Organization.In_Phrase}; {Person, Organization}; Lower; Middle; 5; Alpha>
  SIM<ekibine; ekip+Noun+A3sg+P3sg+Dat; {}; {}; Lower; Middle; 7; Alpha>
  SIM<silahlı; silah+Noun+A3sg+Pnon+Nom+^DB+Adj+With; {Person.Title, Organization.In_Phrase}; {Person, Organization}; Lower; Middle; 7; Alpha>
  SIM<saldırı; saldırı+Noun+A3sg+Pnon+Nom; {}; {}; Lower; Middle; 7; Alpha>
  SIM<düzenlendi; düzenle+Verb+^DB+Verb+Pass+Pos+Past+A3sg; {Organization.In_Phrase}; {Organization}; Lower; Long; 10; Alpha>
  SIM<.; .+Punc; {Person.Short_Title, Person.In_Phrase}; {Person}; Unclass; Short; 1; Punc>]

- Generalized Rule:

  [LOCATED_IN : LOCATION : LOCATION :
  REF_ENT<LOCATION; *+Noun+Prop+A3sg+Pnon+*>
  ENT<LOCATION; *+Noun+Prop+A3sg+Pnon+*>
  *
  SIM<.; .+Punc; {Per.Short_Title, Per.In_Phrase}; {Per}; Unclass; Short; 1; Punc>]

- Recognizable Relations:

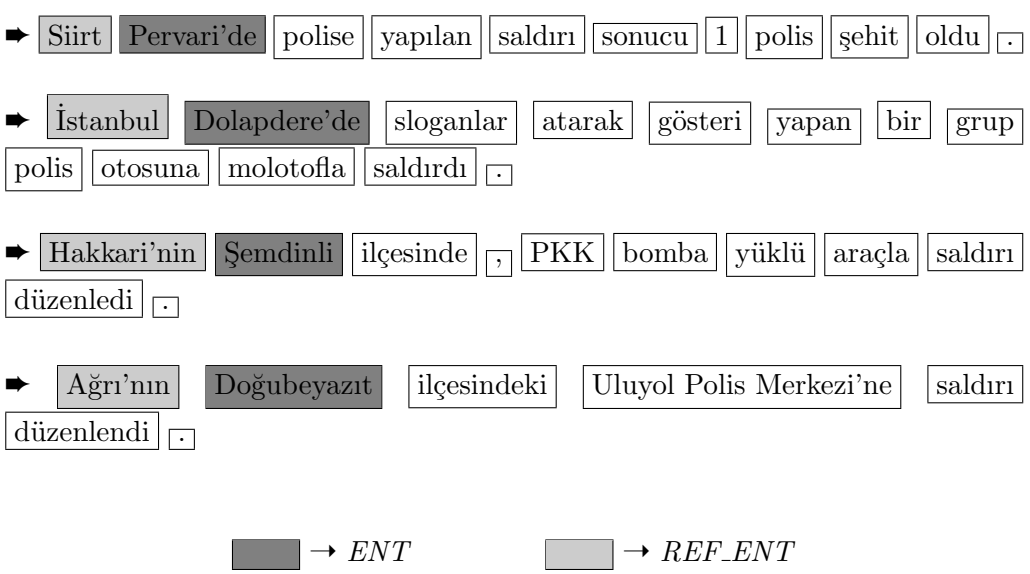


Figure 5.2: An example ERD rule generation

Chapter 6

Experimental Evaluation

We conducted a set of experiments in order to evaluate the performance and the behavior of the proposed methods under different conditions. The main objective of the experimentation is to analyze the performance and the behavior of the methods on realistic data, with different setups. This chapter presents the results of our experimentation and compares the efficiency and accuracy of our approach to several other studies.

6.1 Data

Lavelli et al. discussed the issues specific to IE evaluation that need to be addressed to allow a fair comparison of different systems in [62]. An important issue mentioned in their paper is the description of the used corpus and the exact definition of the corpus partition. In fact, a major obstacle to Turkish IE is the scarcity of publicly available annotated corpora. The experiments for the evaluation of the developed systems are conducted on the TurkIE dataset[1]. In order to generate the dataset, the TurkIE corpus tagging tool was developed. We manually tagged 355 news articles on terrorism from both online and print news sources in Turkish using the developed tagging tool. Annotation process resulted in a single corpus file in XML format. We mainly stuck to the MUC-7 task definitions [17, 45, 18] as guidelines, although we used

[1] http://www.cs.bilkent.edu.tr/ ilyas/TurkIE.rar

a different notation for tagging the articles. The developed notation is easy-to-use for multiple purposes, more readable for human and machines and inherits all features of XML (i.e. simplicity, extensibility, interoperability, openness). Moreover, the resulting corpus can be used for other machine learning tasks. In addition to the named entities and the relations located in the articles, which are the main focus of this study, the text is also splitted into tokens, sentences and topics for other potential uses (e.g. sentence boundary detection, topic segmentation). A sample tagged news article from the corpus is provided in Appendix A.

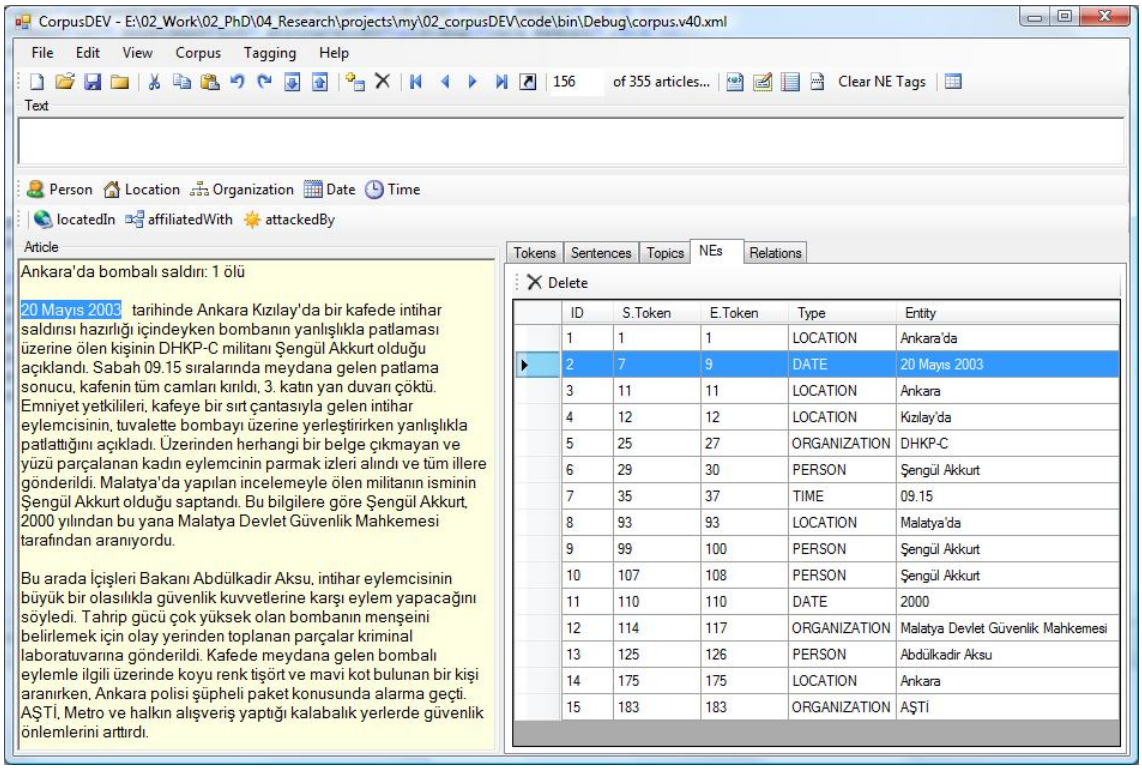


Figure 6.1: TurkIE Corpus Tagger Tool

6.1.1 TurkIE Corpus Tagger

TurkIE corpus tagger is developed to allow the users to tag texts using a graphical interface in a user-friendly environment. A snapshot of the tool is shown in Figure 6.1. The *upper toolbar* provides user the ability to navigate between articles and do some batch processes (e.g. tokenize all articles). The top *text pane* is the section where the user can edit the input text before importing to the corpus. Below the *text pane*,

marking toolbars for named entity and relation tagging are placed. The yellow *article pane* on the bottom leftt of the tagger keeps the original article text in read-only mode. The *tab control* on the bottom right allows user to switch between the *tagged item lists*. The *article pane* and the lists are interactive in the sense that the user can select any tagged item in the lists and the associated part of the text in the *article pane* is highlighted.

6.1.2 Token, Sentence and Topic Tagging

Prior to the named entity and relation annotation, the articles were tokenized, splitted into sentences, and segmented into topics. The tagger provides some automatic utilities for these operations. When used, these utilities parse the article text and automatically detect the boundaries. The user can always make changes on the detected boundaries.

We followed the standard tokenization method which uses white-space and punctuation characters as delimiters except that we removed the apostrophe symbol (') from our delimiter set since morphemes coming after the named entities are considered as part of the name in our study. For TurkIE dataset, token tagging was performed without human intervention. A simple XML element is defined for marking tokens. *Token* element has four attributes:

- *id*: a unique identifier for the token (*int* type)
- *startPos*: starting character position in the text (*int* type)
- *endPos*: ending character position in the text (*int* type)
- *stringValue*: string value of the token (*string* type)

Some examples of the tagged tokens are shown in Figure 6.2.

```
<Token id="1" startPos="0" endPos="9" stringValue="Ankara'da" />

<Token id="7" startPos="36" endPos="38" stringValue="20" />

<Token id="33" startPos="228" endPos="229" stringValue="." />
```

Figure 6.2: Some Examples of the Tagged Tokens

The tagging tool provides a basic capability of splitting text into sentences. It simply divides text from possible sentence boundaries (e.g.“.”, “?”,“!”), without trying to adress sentence boundary detection problem which is not the main objective of this study. For TurkIE dataset, sentence tagging was performed mostly automatically. After automatic sentence tagging, we manually corrected some incorrectly marked sentences. The titles and the timestamps in the text articles are also tagged as sentences. A simple XML element is defined for marking sentences. *Sentence* element has three attributes:

- *id*: a unique identifier for the sentence (*int* type)
- *startToken*: starting token in the text (*int* type)
- *endToken*: ending token in the text (*int* type)

An example tagged sentence is shown in Figure 6.3.

```
<Sentence id="2" startToken="7" endToken="33" />
```

Figure 6.3: An Example Tagged Sentence

The tagging tool provides a basic capability of dividing text into topics. It simply divides text from possible topic boundaries (e.g. titles, subtitles), without trying to adress topic segmentation problem which is not in the scope of this study. For TurkIE dataset, sentence tagging was performed mostly manually. A simple XML element is defined for marking topics. *Topic* element has three attributes:

- *id*: a unique identifier for the topic (*int* type)
- *startToken*: starting token in the text (*int* type)
- *endToken*: ending token in the text (*int* type)

An example tagged topic is shown in Figure 6.4.

```
<Topic id="1" startToken="1" endToken="195" />
```

Figure 6.4: An Example Tagged Topic

6.1.3 Named Entity Tagging

The articles are annotated for three named entity and two temporal expression categories: (1) *PERSON*, (2) *LOCATION*, (3) *ORGANIZATION*, (4) *DATE*, and (5) *TIME*. Since NER task definition has been presented in Section 4.1, this section only covers how named entities are marked by using the developed tagging tool and how they are kept in the corpus file.

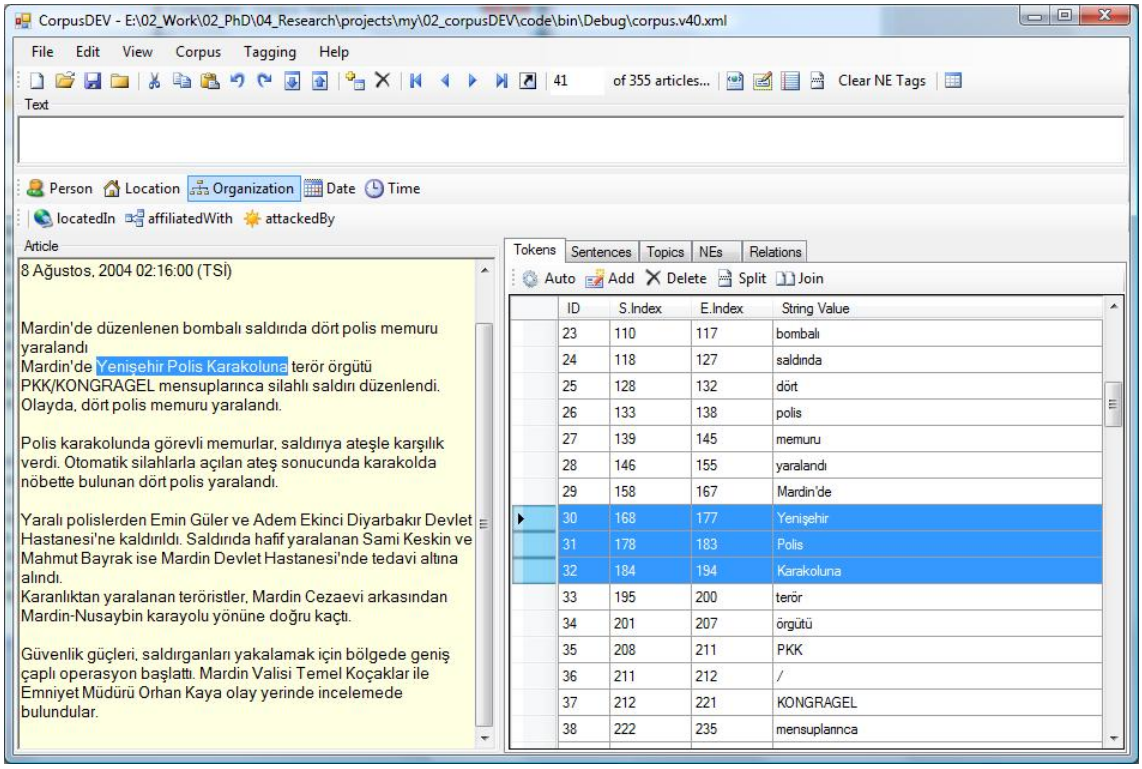


Figure 6.5: Named Entity Tagging in TurkIE Corpus Tagger

Named entity tagging is performed manually. The annotator uses the mouse to select token(s) in the token list and then chooses an appropriate label from the toolbar (Figure 6.5). Five separate simple XML elements with the same attributes (one for each name type) are defined for name tagging. The element name identifies the NE type. NE elements have four attributes:

- *id*: a unique identifier for the named entity (*int* type)
- *startPos*: starting token in the text (*int* type)
- *endPos*: ending token in the text (*int* type)

- *stringValue*: string value of the named entity (*string* type)

Some examples of the tagged named entites are shown in Figure 6.6.

<Date id="2" startToken="7" endToken="9" stringValue="20 Mayıs 2003" />

<Location id="3" startToken="11" endToken="11" stringValue="Ankara" />

<Location id="4" startToken="12" endToken="12" stringValue="Kızılay'da" />

<Time id="8" startToken="11" endToken="18" stringValue="15:39:00 (TSI)" />

<Person id="3" startToken="9" endToken="10" stringValue="Gaffar Okkan'a" />

<Organization id="15" startToken="104" endToken="106" stringValue="Ankara Emniyet Müdürlüğü" />

<Organization id="16" startToken="107" endToken="110" stringValue="Terörle Mücadele Şube Müdürlüğü" />

Figure 6.6: Example Tagged Named Entities

6.1.4 Relation Tagging

Three relation types annotated in the articles are: (1) *LOCATED_IN*, (2) *ATTACKED_BY*, and (3) *AFFILIATED_WITH*. Since ERD task definition has been presented in Section 5.1, this section only covers how relations are marked by using the developed tagging tool and how they are stored in the corpus file.

Relation tagging is performed manually. The relation annotation interface (Figure 6.7) provides the article itself, two lists of the named entities in the article and the token list to the user for tagging. The interactive article text pane is used to view the article text and navigate between the different items in the lists. In order to tag a new relation, the annotator uses the mouse to select the relation type from the top pull-down list, referencing entity from the left named entity list, referenced entity from the middle named entity list, token(s) from which the relation inferred from the token list. The panel, located at the top of the window, provides visual depictions of the defined relations. Three separate complex XML elements with the same attributes (one for each relation type) are defined for relation tagging. The element name identifies the relation type. Relation elements have four attributes:

- *id*: a unique identifier for the relation (*int* type)
- *entityId*: referencing entity number (*int* type)
- *refId*: referenced entity number (*int* type)
- *inferredFromText*: the text piece from which the relation inferred (*string* type)

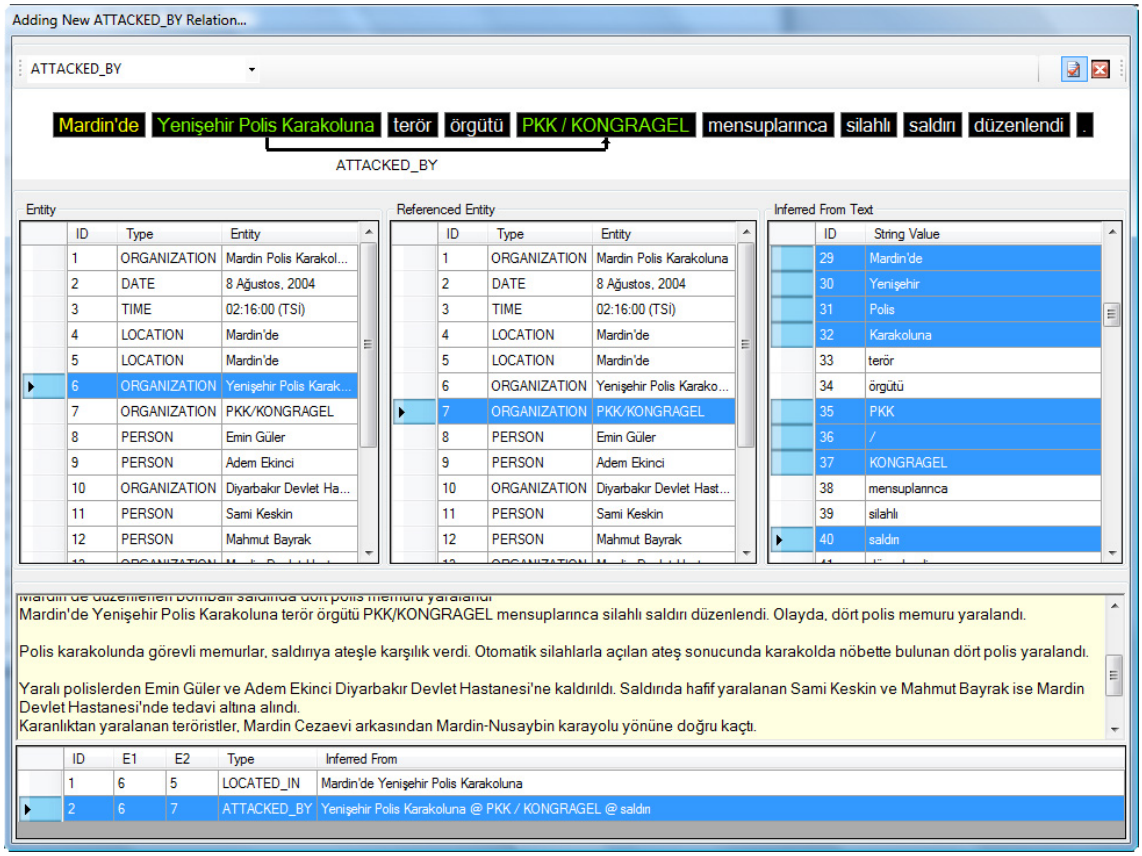


Figure 6.7: Relation Tagging in TurkIE Corpus Tagger

inferredFromText attribute can be a continuous portion of the article or a concatanation of several discrete portions. In the latter case '@' symbol is used to separate different parts of the text. In addition to *inferredFromText* attribute, relation elements contain simple elements representing the tokens covered in the *inferredFromText* attribute. Some examples of the tagged relations are shown in Figure 6.8.

```
<LocatedIn id="1" entityId="6" refId="5" inferredFromText="Bingöl'ün Genç ilçesine">
<InferredFromToken id="1" tokenRef="28" text="Bingöl'ün" />
<InferredFromToken id="2" tokenRef="29" text="Genç" />
<InferredFromToken id="3" tokenRef="30" text="ilçesine" />
</LocatedIn>

<LocatedIn id="3" entityId="7" refId="6" inferredFromText="Genç ilçesine bağlı Yeniyazı Jandarma Karakolu'na">
<InferredFromToken id="1" tokenRef="29" text="Genç" />
<InferredFromToken id="2" tokenRef="30" text="ilçesine" />
<InferredFromToken id="3" tokenRef="31" text="bağlı" />
<InferredFromToken id="4" tokenRef="32" text="Yeniyazı" />
<InferredFromToken id="5" tokenRef="33" text="Jandarma" />
<InferredFromToken id="6" tokenRef="34" text="Karakolu'na" />
</LocatedIn>

<AttackedBy id="8" entityId="7" refId="8" inferredFromText="Yeniyazı Jandarma Karakolu'na terör örgütü PKK / KONGRA - GEL @ ateş açtı">
<InferredFromToken id="1" tokenRef="32" text="Yeniyazı" />
<InferredFromToken id="2" tokenRef="33" text="Jandarma" />
<InferredFromToken id="3" tokenRef="34" text="Karakolu'na" />
<InferredFromToken id="4" tokenRef="35" text="terör" />
<InferredFromToken id="5" tokenRef="36" text="örgütü" />
<InferredFromToken id="6" tokenRef="37" text="PKK" />
<InferredFromToken id="7" tokenRef="38" text="/" />
<InferredFromToken id="8" tokenRef="39" text="KONGRA" />
<InferredFromToken id="9" tokenRef="40" text="-" />
<InferredFromToken id="10" tokenRef="41" text="GEL" />
<InferredFromToken id="11" tokenRef="43" text="ateş" />
<InferredFromToken id="12" tokenRef="44" text="açtı" />
</AttackedBy>

<AffiliatedWith id="12" entityId="22" refId="21" inferredFromText="Hakkari Tugay Komutanlığı'nda görevli @ Şevket Kaygısız">
<InferredFromToken id="1" tokenRef="164" text="Hakkari" />
<InferredFromToken id="2" tokenRef="165" text="Tugay" />
<InferredFromToken id="3" tokenRef="166" text="Komutanlığı'nda" />
<InferredFromToken id="4" tokenRef="167" text="görevli" />
<InferredFromToken id="5" tokenRef="169" text="Şevket" />
<InferredFromToken id="6" tokenRef="170" text="Kaygısız" />
</AffiliatedWith>
```

Figure 6.8: Example Tagged Relations

6.1.5 Corpus Statistics

The The TurkIE corpus contains 54499 tokens, 3552 sentences and 558 topics. 5692 named entities were tagged in 5 categories: 1336 *PERSON* names, 2338 *LOCATION* names, 1249 *ORGANIZATION* names, 376 *DATE* expressions and 393 *TIME* expressions. 921 relations were tagged in 3 categories: 99 *ATTACKED_BY* relations, 719 *LOCATED_IN* relations, 103 *AFFILIATED_WITH* relations.

6.2 Methodology

Another issue discussed in [62] is how tolerantly to assess inexact identification of NE boundaries. Among three different criteria - exact, contains, and overlap - discussed in [62] for matching reference instances and extracted instances, we used the exact criteria which is the most conservative approach to determine the truth-value of the matching. In our experiments, a predicted instance is not considered as a correct match unless it matches exactly an actual instance in the text. In our scoring, the expectation from the system is to find all occurrences of the named entities. However, we restrict relation detection to finding relationships between entities in the same sentence.

In order to evaluate the developed methods, 10-fold cross validation was performed on the dataset. We measured precision, recall, and F-score; as is commonly done in the Message Understanding Conference (MUC) evaluations. Precision is the fraction of correct outcomes divided by the number of all outcomes. For instance, precision value for the NER task is the percentage of extracted named entities that are correct. On the other hand, recall is analogous to sensitivity in binary classification. Recall can be defined as the fraction of correct outcomes divided by the total number of possible correct answers. The F-score, harmonic mean of precision and recall, provides a method for combining precision and recall scores into a single value.

NE Category	Precision (%)	Recall (%)	F-Score (%)
Person	92.08	96.69	94.33
Location	89.86	90.20	90.03
Organization	88.01	87.36	87.68
Date	95.34	97.69	96.50
Time	91.00	93.12	92.05
Overall	90.43	91.74	91.08

Table 6.1: Quantitative performance results of the developed NER system

The last row shows the overall extraction performance of the developed system. We use the standard formula for precision, recall, and F-score calculation: precision = (true positives)/(true positives + false positives); recall = (true positives)/(true positives + false negatives); F-score = (2 * precision * recall)/(precision + recall).

6.3 Results & Discussion

6.3.1 Named Entity Recognition

This section presents the performance and the behavior of the developed NER system.

6.3.1.1 Quantitative Results & Comparison of the Methods

Table 6.1 shows the quantitative results of the experiments performed. The developed NER system achieved overall performance of *F=91.08%* on the dataset. The system reached best performance score *F=96.5%* on locating *DATE* fields; the system extracted *97.69%* of the *DATE* fields in the test dataset and *95.34%* of the found *DATE* fields were correct. It achieved *F=87.68%* on a more challenging NE type, *ORGANIZATION*.

The system produced better results than many of the previous studies. One important point to highlight before discussing the results of the comparisons is that the comparisons were not made on the same datasets, since the datasets were not publicly available. Conducting their experiments on a relatively small corpus, Cucerzan and Yarowsky [23] reported *F=53.04%* for NER from Turkish texts. They aimed at

building a maximally language-independent system for named-entity recognition and classification, which lead using minimal information about the source language. They used an EM-style bootstrapping algorithm based on iterative learning and re-estimation of contextual and morphological patterns captured in hierarchically smoothed trie models. With a focus on only Turkish, our algorithm is designed to take advantage of the specific characteristics of the Turkish language and to address any challenges pertaining to the language. Moreover, our algorithm is capable of utilizing several resources (e.g. dictionaries and gazetteer lists) to obtain better extraction performance.

Bayraktar and Temizel [9] reported *F=81.97%* for person name extraction from Turkish text using local grammar approach. They focused on reporting verbs (e.g. said, told) in the sentences, and used these reporting verbs as the clues for extracting person names. Their study covered finding significant reporting verbs in Turkish and obtaining hand-crafted extraction patterns by conducting concordance analysis by using the found forms of reporting verbs. The rule-based system developed by Kucuk and Yazici [57] achieved *F= 78.7%* for the task of NER from Turkish news articles. Their system heavily relies on a manually compiled set of lexical resources (e.g. dictionary of person names, list of well-known people) and hand-crafted pattern bases to extract each named entity type in its scope. In order to mitigate the issues caused by the agglutinative nature of Turkish, they used a morphological analyzer. The system is designed to extract the phrases which exist in the provided lexical resources, conform to the patterns in the pattern bases or inflectional forms of those. Both of the systems presented in [9] and [57] depend heavily on the lexical resources and the manually developed rules/patterns. One drawback of these systems is the high human expertise required for the development and management of the rules. Another well-known shortcoming is their adaptability to new domains. For instance, Kucuk and Yazici [57] reported that their system scored *F=69.3%* and *F=55.3%* on *Child Stories* and *Historical Text* domains respectively.

The statistical learning system presented by Tur et al. [101] reached *F=91.56%*. When the learning strategy and the information sources used are considered, their system is the most similar previous work. Both our approach and theirs use supervised learning strategy. Furthermore, both methods exploit similar features (lexical, morphological, contextual, etc.), though there are differences in the ways how the features are utilized.

6.3.1.2 Error Analysis

The analysis conducted revealed that the extraction errors are more frequently occur in the following cases:

- *Nested NEs*: Nested NE constructs are common forms observed in texts. Following MUC-7 named entity task definition [17], only the longest NE was tagged in case of nested NEs. For instance, *"5 Ocak Caddesi"* (*5th January Street*) is tagged as one location name: *<LOCATION>5 Ocak Caddesi</LOCATION>*, instead of a *DATE* and a *LOCATION* name: *<LOCATION><DATE>5 Ocak</DATE> Caddesi</LOCATION>*. Recognizing the inner NE (e.g. *"5 Ocak"*), and missing the outer one (e.g. *"5 Ocak Caddesi"*) is observed as one type of the erroneous extractions.
- *Long NEs*: Partial detection of the long NEs, especially long *ORGANIZATION* and *LOCATION* names, is another frequent type of the erroneous extractions. The analysis showed that the average system performance for the long NEs is below the overall performance of the system.

6.3.1.3 Threshold Factor

The coverage algorithm has a user-set threshold parameter which has an impact on the performance of our extraction method. Figure 6.9 shows the performance of the NER system as the threshold changes. The optimum value for the threshold parameter is found to be *0.87*, where the F-score is maximized, through experimentation.

In the first half of the graph, we observe a continuous climbing trend in the precision and recall at the same time. The increase in the precision parallel to the increase in the threshold value is a normal behaviour. However, one would expect inversely proportional relation between the recall ad the threshold. The observed situation is due to the fact that the longest candidate among the conflicting candidate phrases is selected during the post-processing step. In the second half, as expected, the recall value decreases and the precision value increases with the increase in threshold. Another notable observation is the local drops in the recall rate where the threshold parameter is *0.5*. This behavior is caused by the elimination of a general rule whose true positive

(TP) returns are more than its false positive (FP) returns. A similar situation occurs where the threshold parameter is *0.8*.

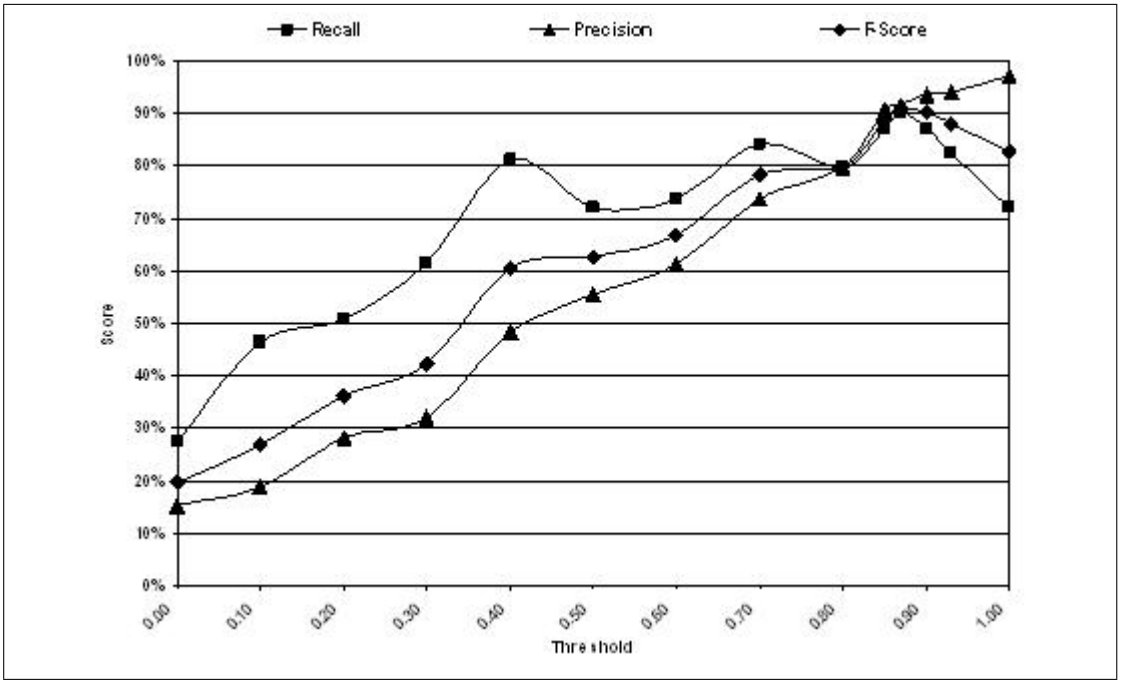


Figure 6.9: The observed performance of the developed NER system as the threshold parameter changes

6.3.1.4 Generalization Features

We investigated the impacts of each feature set used for the generalization process. In order to calculate individual contribution of each set, we conducted two series of experiments. First, we deactivated a feature set in each experiment, and recorded the achieved system performance. Second, we approached the question from a different point of view. This time, we tested the system performance by using only one feature set at a time. Table 6.2 and Table 6.3 show the results of two experiment series conducted. Table 6.2 shows the recorded performance score and the incurred performance loss in the absence of each specific generalization feature. The biggest loss occurs (*12.26%*) when morphological features were not used. Individual impact of each feature set to the system performance is shown in a different way in Table 6.3; the recorded performance scores and the incurred performance losses were given when only one feature set and the actual token information were used at a time. The system achieved *F=69.95%* using only morphological features and the actual token information.

Deactivated Feature Set	**Used Feature Sets**	**System Performance (F-Score (%))**	**F-Loss (%)**
Lexical	Morphological, Contextual, Orthographic	81.29	9.79
Morphological	Lexical , Contextual, Orthographic	***78.82***	***12.26***
Contextual	Lexical , Morphological, Orthographic	87.90	3.18
Orthographic	Lexical, Morphological, Contextual	81.71	9.37

Table 6.2: Individual impact of each feature set to the developed NER system performance (I).

Each time a feature set was deactivated and the achieved system performance value was recorded. The last column shows the performance loss incurred when the specific set was not used. The loss incurred in this series is directly proportional to the impact of the deactivated feature set.

Used Feature Set	**Deactivated Feature Sets**	**System Performance (F-Score (%))**	**F-Loss (%)**
Lexical	Morphological, Contextual, Orthographic	67.21	23.87
Morphological	Lexical , Contextual, Orthographic	***69.95***	***21.13***
Contextual	Lexical , Morphological, Orthographic	56.51	34.57
Orthographic	Lexical, Morphological, Contextual	65.15	25.93

Table 6.3: Individual impact of each feature set to the developed NER system performance (II).

Only one feature set was used at a time. The last column shows the performance loss incurred when only that specific set and the actual token information were used. The loss incurred in this series is inversely proportional to the impact of the used feature set.

6.3.1.5 Automatic Rule Learning for Protein Name Extraction

We used an adapted version of the developed NER system for protein name extraction from biological texts [95]. This system use the same coverage algorithm and similar pattern elements. Protein names are generalized by using hierarchically categorized syntactic token types. The experiments were conducted on two different datasets: the YAPEX corpora [28] and the GENIA corpus [55]. The performance scores on the YAPEX dataset were recorded both with and without cross validation for comparison purposes. Ten-fold cross validation were performed on the GENIA dataset to evaluate the developed method. Altough the protein name recognizer was an early version with small number of variables defined for generalization and the granularity level of the protein name extraction rules were not as detailed as the rules learnt for Turkish, it achieved satisfying results: 61.8% F-score value on the YAPEX dataset and 61.0% on the GENIA corpus. This results show that the system is effective for protein name extraction and achieved better performance than some of the previous work.

6.3.2 Entity Relation Detection

This section presents the performance and the behavior of the developed ERD system.

6.3.2.1 Quantitative Results

Table 6.4 shows the quantitative results of the experiments performed. The developed ERD system achieved overall performance of *F=71.68%* on the dataset. The system reached best performance score *F=78.57%* on locating *LOCATED_IN* fields; the system extracted *75.34%* of the *LOCATED_IN* fields in the test dataset and *82.09%* of the found *LOCATED_IN* fields were correct. It achieved *F=46.15%* on *ATTACKED_BY*, and *F=40.00%* on *AFFILIATED_WITH* categories.

We believe that system's better performance on the extraction of *LOCATED_IN* relation is a result of the fact that the number of *LOCATED_IN* relations in the dataset is by far more than the number of the other two relation types.

Relation Category	Precision (%)	Recall (%)	F-Score (%)
LOCATED_IN	82.09	75.34	78.57
ATTACKED_BY	75.00	33.33	46.15
AFFILIATED_WITH	100	25.00	40.00
Overall	82.67	63.27	71.68

Table 6.4: Quantitative performance results of the developed ERD system

The last row shows the overall extraction performance of the developed system. We use the standard formula for precision, recall, and F-score calculation: precision = (true positives)/(true positives + false positives); recall = (true positives)/(true positives + false negatives); F-score = (2 * precision * recall)/(precision + recall).

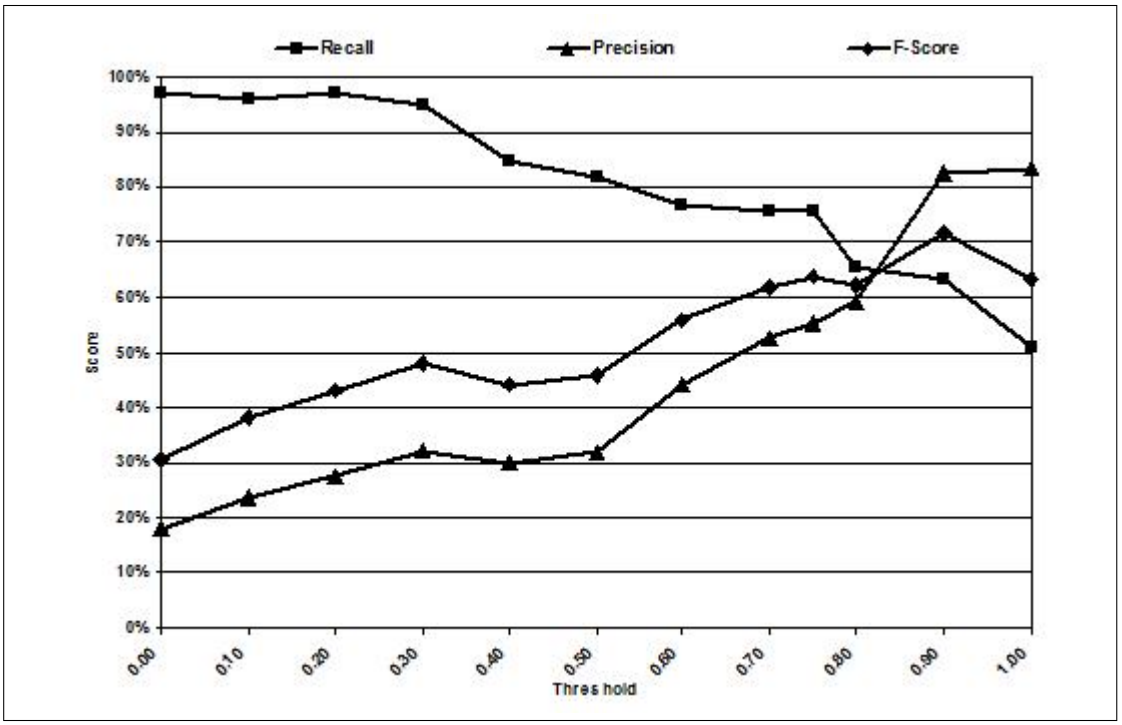


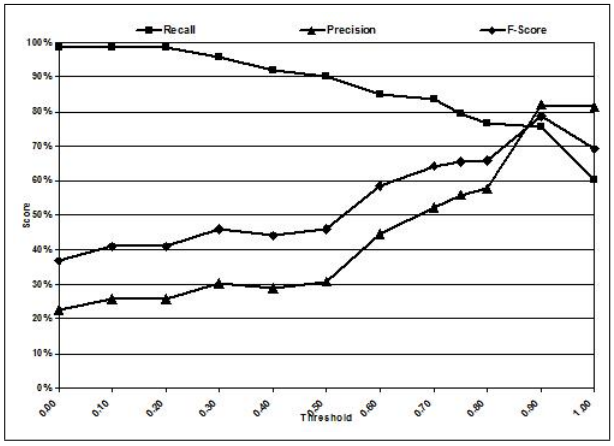
Figure 6.10: The observed performance of the developed ERD system as the threshold parameter changes

6.3.2.2 Threshold Factor

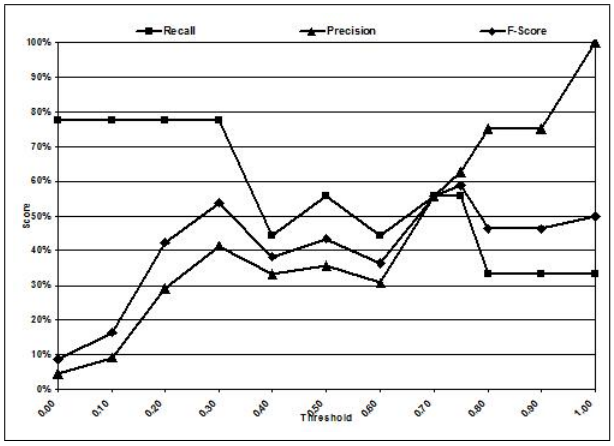
The threshold parameter controls the trade-off between precision and recall scores. Figure 6.10 shows the performance of the ERD system as the threshold parameter changes. The graph shows that the system shows the expected behaviour; the recall value decreases and the precision value increases with the increase in threshold. The optimum value for the threshold parameter is found to be *0.90*, where the acquired F-score value is maximized, through experimentation.

Another important point to underline is that diferent thresholds values can maximize F-score values for different relation categories. For instance, the optimum value for the threshold parameter is found to be *0.90* for *LOCATED_IN* relations. It is *0.75* for *ATTACKED_BY* and *AFFILIATED_WITH*. Figure 6.11 shows the performance of the ERD system for different relation categories as the threshold parameter changes.

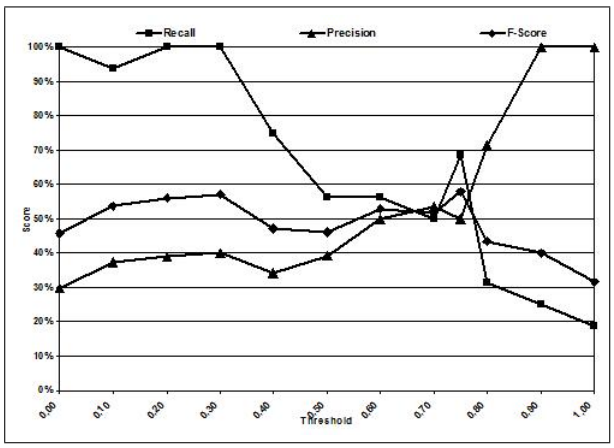
The system performance for *LOCATED_IN* at different threshold values is shown in Figure 6.11(a). It is obvious that the *LOCATED_IN* performance dominates the overall system performance, since the number of *LOCATED_IN* relations is more than the number of the other two relation types. The system shows a similar performance behaviour for *ATTACKED_BY* and *AFFILIATED_WITH* relation types, as seen in Figure 6.11(b) and Figure 6.11(c).



(a) LOCATED_IN



(b) ATTACKED_BY



(c) AFFILIATED_WITH

Figure 6.11: The observed performance of the developed ERD system for different relation categories as the threshold parameter changes

Chapter 7

Conclusion

The research presented in this thesis has focused on the design and evaluation of automatic rule learning methods for two important tasks in information extraction field: named entity recognition, and entity relation detection. The presented methods aim to learn rules automatically to recognize the patterns available in the texts and generalize these patterns by processing similarities and differences between them.

We have first described the developed method for identifying NEs located in Turkish texts. The system utilized several generalization features to obtain accurate generalization and remedy the issues related to the data sparseness problem. In addition to the generalization features, an expressive rule representation language and a novel coverage algorithm are used by the system for automatic rule learning and generalization. We performed several experiments to evaluate the performance of our method. The results indicate that our suggested method can be used for extracting NEs from Turkish texts effectively. Although there are a few studies available yet, we compared the performance of the developed system with the previous studies. Our system produced better results than many of the previous studies. The impact of each generalization feature utilized for NER was investigated from different angles. The results show that exploiting morphological features significantly improves the NER from Turkish texts due to the agglutinative nature of the language. We believe that the use of morphological features can improve the NER performance in other agglutinative languages too.

The second method presented in the thesis uses the same concepts to detect relationships between the named entities in Turkish texts. To our knowledge, our study is the first to examine ERD task in Turkish. The experiments demonstrated that we have succesfully applied the developed model to the task. Unfortunately, the lack of studies addressing ERD in Turkish prevented us comparing our results with others.

Both of the methods do not heavily suffer from domain adaptation problem, another key challenge in the IE field, by employing an adaptive rule learning method. The developed systems minimize the tasks requiring human intervention; it does not rely on manually developed rules/patterns. The lexical sources used in the systems are kept generic to capture the variations in the patterns. The system also eliminates the burden of adding new sources by its configurable and extensible design. Moreoever, an adapted version of the automatic rule learning algorithm is applied to protein name extraction task and achieved satisfying results.

The lack of defined task definitions and training data for Turkish, and limited number of studies focused on Turkish are the main issues we had to deal with. Another contribution of this study lies in the adaptation of the NER and ERD task definitions to Turkish. We generally followed the existing MUC-7 task definitions [17, 45, 18], but distinct characteristics of Turkish led us to make several adaptations to these definitions. The experiments were conducted on the TurkIE corpus, generated in support of this study. The developed corpus and the annotation tool are two other major contributions of this study, which will encourage and support future researchers in this area.

We think that there is still plenty of room for further research in IE from Turkish texts. Our future plans include improving the developed named entity recognizer and entity relation detector to further increase the performance and decrease the training time. One research direction would be the introduction of correction rules to the learning method. Although the use of rule exception sets helps reducing false positives, handling the information regarding to the rule exceptions in a more formal way and generalizing them into correction rules would provide further increase in the performance of the system. The current relation extractor identifies and extracts relationships at the sentence level. We are planning to extend the relation extractor to detect relations that span multiple sentences. Another area of future work is to further expand the TurkIE corpus in order to examine the behavior of the developed systems on larger datasets. Moreover, we would like to see the system's behaviour in

the presence of noise. Generation of noisy data for test purposes will also be taken into consideration during the corpus expansion study.

Bibliography

[1] Proceedings of the fifth message understanding conference. Morgan Kaufmann, 1993.

[2] Proceedings of the fourth message understanding conference. Morgan Kaufmann, 1992.

[3] Proceedings of the seventh message understanding conference. 1998.

[4] Proceedings of the sixth message understanding conference. Morgan Kaufmann, 1995.

[5] Proceedings of the third message understanding conference. Morgan Kaufmann, 1991.

[6] E. Alfonseca and S. Manandhar. An unsupervised method for general named entity recognition and automated concept discovery. In *Poceedings of the First International Conference on General WordNet*, Mysore, India, 2002.

[7] Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, David Martin, Karen Myers, and Mabry Tyson. SRI international FASTUS system: MUC -6 test results and analysis. In *MUC 6 '95: Proceedings of the 6th conference on Message understanding*, pages 237–248, Morristown, NJ, USA, 1995. Association for Computational Linguistics.

[8] Arvind Arasu. Extracting structured data from web pages. In *In ACM SIGMOD*, pages 337–348, 2003.

[9] Ozkan Bayraktar and Tugba Taskaya Temizel. Person name extraction from Turkish financial news text using local grammar based approach. In *23rd*

International Symposium on Computer and Information Sciences (ISCIS'08), 10/2008 2008.

[10] Yassine Benajiba, Mona Diab, and Paolo Rosso. Arabic named entity recognition using optimized feature sets. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 284–293, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

[11] E. Bick. A named entity recognizer for danish. In *Proceedings of the Conference on Language Resources and Evaluation*, 2004.

[12] Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Mach. Learn.*, 34(1-3):211–231, 1999.

[13] S. Boutsis, I. Demiros, V. Giouli, M. Liakata, H. Papageorgiou, and S. Piperidis. A system for recognition of named entities in greek. In *Proceedings of the International Conference on Natural Language Processing*, 2000.

[14] Mary Elaine Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 328–334, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[15] Mary Elaine Califf and Raymond J. Mooney. Bottom-up relational learning of pattern matching rules for information extraction. *J. Mach. Learn. Res.*, 4:177–210, 2003.

[16] Hai Leong Chieu and Hwee Tou Ng. A maximum entropy approach to information extraction from semi-structured and free text. In *Eighteenth national conference on Artificial intelligence*, pages 786–791, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[17] Nancy Chinchor. MUC -7 named entity task definition, version 3.5. In *Proceedings of the Seventh Message Understanding Conference*, 1998.

[18] Nancy Chinchor and Elaine Marsh. MUC -7 information extraction task definition, version 5.1. In *Proceedings of the Seventh Message Understanding Conference*, 1998.

[19] Euisok Chung, Yi-Gyu Hwang, and Myung-Gil Jang. Korean named entity recognition using HMM and cotraining model. In *Proceedings of the sixth international workshop on Information retrieval with Asian languages - Volume 11*, pages 161–167, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[20] Ilyas Cicekli and Nihan Kesim Cicekli. Generalizing predicates with string arguments. *Applied Intelligence*, 25(1):23–36, 2006.

[21] Science Applications International Corporation. Science applications international corporation (SAIC) information extraction website. Retrieved September, 2010 from http://www.itl.nist.gov/iaui/894.02/related_projects/muc/.

[22] Alessandro Cucchiarelli and Paola Velardi. Unsupervised named entity recognition using syntactic and semantic contextual evidence. *Comput. Linguist.*, 27(1):123–131, 2001.

[23] Silviu Cucerzan and David Yarowsky. Language independent named entity recognition combining morphological and contextual evidence. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 90–99, 1999.

[24] Turhan Daybelge and Ilyas Cicekli. A rule-based morphological disambiguator for Turkish. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2007), Borovets*, pages 145–149, 2007.

[25] D. Downey, O. Etzioni, D. S. Weld, and S. Soderland. Learning text patterns for web information extraction and assessment. In *Proceedings of the AAAI-04 Workshop on Adaptive Text Extraction and Mining*, 2004.

[26] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165(1):91–134, 2005.

[27] Aidan Finn and Nicholas Kushmerick. Multi-level boundary classification for information extraction. In *European Conference on Machine Learning (ECML)*, pages 111–122, 2004.

[28] Kristofer Franzén, Gunnar Eriksson, Fredrik Olsson, Lars Asker, Per Lidén, and Joakim Cöster. Protein names and how to find them. *I. J. Medical Informatics*, 67(1-3):49–61, 2002.

[29] Dayne Freitag. Information extraction from HTML: application of a general machine learning approach. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, pages 517–523, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

[30] Dayne Freitag. Machine learning for information extraction in informal domains. *Mach. Learn.*, 39(2-3):169–202, 2000.

[31] Dayne Freitag and Nicholas Kushmerick. Boosted wrapper induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 577–583. AAAI Press, 2000.

[32] Dayne Freitag and Andrew McCallum. Information extraction with HMM structures learned by stochastic optimization. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 584–589. AAAI Press, 2000.

[33] Dayne Freitag and Andrew Kachites Mccallum. Information extraction with HMMs and shrinkage. In *In Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 31–36, 1999.

[34] Dayne Brian Freitag. *Machine learning for information extraction in informal domains*. PhD thesis, Pittsburgh, PA, USA, 1999.

[35] Guohong Fu and Kang-Kwong Luke. Chinese named entity recognition using lexicalized HMMs. *SIGKDD Explor. Newsl.*, 7:19–25, 2005.

[36] K. Fukuda, T. Tsunoda, A. Tamura, and T. Takagi. Toward information extraction: Identifying protein names from biological papers. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 707–718, 1998.

[37] Ralph Grishman. The NYU system for MUC -6 or where's the syntax? In *MUC 6 '95: Proceedings of the 6th conference on Message understanding*, pages 167–175, Morristown, NJ, USA, 1995. Association for Computational Linguistics.

[38] Ralph Grishman. Information extraction: Techniques and challenges. In *SCIE '97: International Summer School on Information Extraction*, pages 10–27, London, UK, 1997. Springer-Verlag.

[39] Ralph Grishman and Beth Sundheim. Message Understanding Conference-6: a brief history. In *Proceedings of the 16th Conference on Computational Linguistics*, pages 466–471, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

[40] C. Grover, S. McDonald, D. N. Gearailt, V. Karkaletsis, D. Farmakiotou, G. Samaritakis, G. Petasis, M. T. Pazienza, M. Vindigni, F. Vichot, and F. Wolinski. Multilingual XML-based named entity recognition for e-retail domains. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, 2002.

[41] Jorge Hankamer. Morphological parsing and the lexicon. In W. Marslen-Wilson, editor, *Lexical Representation and Process*, pages 392–408, Cambridge, MA, 1989. MIT Press.

[42] Zellig Sabbettai Harris. Linguistic transformations for information retrieval. In *International Conference on Scientific Information (1958)*, page 158, 1959.

[43] Kazi Saidul Hasan, Altaf ur Rahman, and Vincent Ng. Learning-based named entity recognition for morphologically-rich, resource-scarce languages. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 354–362, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[44] Marti A. Hearst. Untangling text data mining. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 3–10, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[45] Lynette Hirschman and Nancy Chinchor. MUC -7 coreference task definition, version 3.0. In *Proceedings of the Seventh Message Understanding Conference*, 1998.

[46] Lynette Hirschman, Ralph Grishman, and Naomi Sager. From text to structured information: automatic processing of medical reports. In *AFIPS '76: Proceedings*

of the June 7-10, 1976, National Computer Conference and Exposition, pages 267–275, New York, NY, USA, 1976. ACM.

[47] Jerry R. Hobbs, John Bear, David Israel, and Mabry Tyson. FASTUS : A finite-state processor for information extraction from real-world text. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 1172–1178, 1993.

[48] Chun-nan Hsu and Ming-Tzung Dung. Generating finite-state transducers for semistructured data extraction from the web. *J. Information Systems*, 23(8):521–538, 1998.

[49] Dawei Hu, Huan Li, Tianyong Hao, Enhong Chen, and Liu Wenyin. Heuristic learning of rules for information extraction from web documents. In *Proceedings of the 2nd international conference on Scalable information systems*, InfoScale '07, pages 61:1–61:7, ICST, Brussels, Belgium, Belgium, 2007.

[50] Scott B. Huffman. Learning information extraction patterns from examples. In *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pages 246–260, London, UK, 1996. Springer-Verlag.

[51] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, and Y. Wilks. University of sheffield: Description of the Lasie-II system as used for MUC -7. In *Proceedings of the Seventh Message Understanding Conferences*. Morgan, 1998.

[52] Hideki Isozaki. Japanese named entity recognition based on a simple rule generator and decision tree learning. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 314–321, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[53] Rosie Jones, Rayid Ghani, Tom Mitchell, and Ellen Rilo. Active learning for information extraction with multiple view feature sets. In *IN PROCEEDINGS OF THE ECML-2004 WORKSHOP ON ADAPTIVE TEXT EXTRACTION AND MINING (ATEM-2003*, 2003.

[54] S. Katrenko and P. Adriaans. Named entity recognition for Ukrainian: a resource-light approach. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing: Information Extraction and Enabling Technologies*, pages 88–93, 2007.

[55] J.D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. Genia corpus: a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Suppl 1):180–182, 2003.

[56] Jun-Tae Kim and Dan I. Moldovan. Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE Trans. on Knowl. and Data Eng.*, 7:713–724, October 1995.

[57] Dilek Kucuk and Adnan Yazici. Named entity recognition experiments on Turkish texts. In *Proceedings of the 8th International Conference on Flexible Query Answering Systems*, FQAS '09, pages 524–535, Berlin, Heidelberg, 2009. Springer-Verlag.

[58] N. Kushmerick, D. S. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, volume 1*, pages 729–735, 1997.

[59] Nicholas Kushmerick. Wrapper induction: efficiency and expressiveness. *Artif. Intell.*, 118(1-2):15–68, 2000.

[60] Nicholas Kushmerick and Bernd Thomas. Intelligent information agents. chapter Adaptive information extraction: core technologies for information agents, pages 79–103. Springer-Verlag, Berlin, Heidelberg, 2003.

[61] Mucahit Kutlu. Noun phrase chunker for Turkish using dependency parser. M.S Thesis, Bilkent University, 2010.

[62] A. Lavelli, M. E. Califf, F. Ciravegna, D. Freitag, C. Giuliano, N. Kushmerick, and L. Romano. IE evaluation: Criticisms and recommendations. In *Proceedings of the Workshop on Adaptive Text Extraction and Mining (AAAI-2004)*, 2004.

[63] Anita Louis, Alta De Waal, and Cobus Venter. Named entity recognition in a south african context. In *Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, SAICSIT '06, pages 170–179. South African Institute for Computer Scientists and Information Technologists, 2006.

[64] Peter Lyman and Hal R. Varian. How much storage is enough? *ACM Queue*, 1(4), 2003.

[65] Diana Maynard, Valentin Tablan, Cristian Ursu, Hamish Cunningham, and Yorick Wilks. Named entity recognition from diverse text types. In *In Recent Advances in Natural Language Processing 2001 Conference, Tzigov Chark*, 2001.

[66] Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[67] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 188–191, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[68] Sven Mika and Burkhard Rost. Protein names precisely peeled off free text. *Bioinformatics*, 20:241–247, January 2004.

[69] Einat Minkov, Richard C. Wang, and William W. Cohen. Extracting personal names from email: applying named entity recognition to informal text. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 443–450, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[70] Ion Muslea, Steve Minton, and Craig Knoblock. A hierarchical approach to wrapper induction. In *Proceedings of the Third Annual Conference on Autonomous Agents*, AGENTS '99, pages 190–197, New York, NY, USA, 1999. ACM.

[71] Ion Muslea, Steven Minton, and Craig A. Knoblock. Selective sampling with redundant views. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 621–626. AAAI Press, 2000.

[72] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January 2007.

[73] National Institute of Standards and Technology (NIST). Automatic Content Extraction (ACE) evaluation website. Retrieved September, 2010 from http://www.itl.nist.gov/iad/mig/tests/ace/.

[74] National Institute of Standards and Technology (NIST). Text Analysis Conference (TAC) website. Retrieved September, 2010 from http://www.nist.gov/tac.

[75] Kemal Oflazer. Two-level description of Turkish morphology. In *Proceedings of the sixth conference on European chapter of the Association for Computational Linguistics*, pages 472–472, Morristown, NJ, USA, 1993. Association for Computational Linguistics.

[76] Fuchun Peng and Andrew McCallum. Accurate information extraction from research papers using conditional random fields. In *HLT-NAACL04*, pages 329–336, 2004.

[77] Georgios Petasis, Frantz Vichot, Francis Wolinski, Georgios Paliouras, Vangelis Karkaletsis, and Constantine D. Spyropoulos. Using machine learning to maintain rule-based named-entity recognition and classification systems. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 426–433, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[78] B. Popov, A. Kirilov, D. Maynard, and D. Manov. Creation of reusable components and language resources for named entity recognition in russian. In *Proceedings of the Conference on Language Resources and Evaluation*, 2004.

[79] Adam Przepiórkowski. Slavonic information extraction and partial parsing. In *ACL '07: Proceedings of the Workshop on Balto-Slavonic Natural Language Processing*, pages 1–10, Morristown, NJ, USA, 2007. Association for Computational Linguistics.

[80] Ellen Riloff. Automatically constructing a dictionary for information extraction tasks. In *In Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 811–816. MIT Press, 1993.

[81] Ellen Riloff. An empirical study of automated dictionary construction for information extraction in three domains. *Artif. Intell.*, 85(1-2):101–134, 1996.

[82] Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 474–479, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[83] Marc Rössler. Corpus-based learning of lexical resources for german named entity recognition. In *LREC '04: Proceedings of the Conference on Language Resources and Evaluation*, 2004.

[84] Naomi Sager, Carol Friedman, and Margaret S. Lyman. *Medical Language Processing: Computer Management of Narrative Data*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1987.

[85] K. Saito and M. Nagata. Multi-language named-entity recognition system based on HMM. In *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-Language Named Entity Recognition - Volume 15*, pages 41–48, 2003.

[86] Satoshi Sekine, Ralph Grishman, and Hiroyuki Shinnou. A decision tree method for finding and classifying names in japanese texts. In *In Proceedings of the Sixth Workshop on Very Large Corpora*, 1998.

[87] Kristie Seymore, Andrew Mccallum, and Roni Rosenfeld. Learning hidden markov model structure for information extraction. In *AAAI 99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.

[88] Yusuke Shinyama and Satoshi Sekine. Named entity discovery using comparable news articles. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 848, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[89] Christian Siefkes and Peter Siniakov. An overview and classification of adaptive approaches to information extraction. In *JOURNAL ON DATA SEMANTICS, IV:172212. LNCS 3730*. Springer, 2005.

[90] Joaquim F. Ferreira Da Silva, Zornitsa Kozareva, Jos Gabriel, and Pereira Lopes. Cluster analysis and classification of named entities. In *Proc. Conference on Language Resources and Evaluation*, 2004.

[91] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Mach. Learn.*, 34:233–272, February 1999.

[92] Stephen Soderland, David Fisher, Jonathan Aseltine, and Wendy Lehnert. CRYSTAL inducing a conceptual dictionary. In *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*, pages 1314–1319, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[93] L. Tanabe and W. J. Wilbur. Tagging gene and protein names in biomedical text. *Bioinformatics*, 18(8):1124–1132, August 2002.

[94] Serhan Tatar and Ilyas Cicekli. Automatic rule learning exploiting morphological features for named entity recognition in turkish. *Journal of Information Science*. (in Press).

[95] Serhan Tatar and Ilyas Cicekli. Two learning approaches for protein name extraction. *J. of Biomedical Informatics*, 42:1046–1055, December 2009.

[96] Pham Thi Xuan Thao, Tran Quoc Tri, Dinh Dien, and Nigel Collier. Named entity recognition in vietnamese using classifier voting. *ACM Transactions on Asian Language Information Processing (TALIP)*, 6:1–18, December 2007.

[97] Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. Active learning for natural language parsing and information extraction. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 406–414, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[98] Erik F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: language-independent named entity recognition. In *proceedings of the 6th conference on Natural language learning - Volume 20*, COLING-02, pages 1–4, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[99] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, pages 142–147, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[100] Yoshimasa Tsuruoka and Jun'ichi Tsujii. Improving the performance of dictionary-based approaches in protein name recognition. *J. of Biomedical Informatics*, 37:461–470, December 2004.

[101] Gokhan Tur, Dilek Hakkani-Tur, and Kemal Oflazer. A statistical information extraction system for Turkish. *Natural Languge Engineering*, 9:181–210, June 2003.

[102] Jordi Turmo, Alicia Ageno, and Neus Català. Adaptive information extraction. *ACM Comput. Surv.*, 38(2):4, 2006.

[103] Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, UAI '04, pages 593–601, Arlington, Virginia, United States, 2004. AUAI Press.

[104] Youzheng Wu, Jun Zhao, Bo Xu, and Hao Yu. Chinese named entity recognition based on multiple features. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 427–434, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[105] Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th conference on Computational linguistics*, pages 940–946, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

[106] Roman Yangarber, Lauri Jokipii, Antti Rauramo, and Silja Huttunen. Extracting information about outbreaks of infectious epidemics. In *Proceedings of HLT/EMNLP on Interactive Demonstrations*, pages 22–23, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[107] H. P. Zhang, Q. Liu, H.K. Yu, Y.Q. Cheng, and S. Bai. Chinese named entity recognition using role model. *International Journal of Computational Linguistics and Chinese language processing*, 8:29–60, 2003.

[108] Shubin Zhao and Ralph Grishman. Extracting relations with integrated information using kernel methods. In *ACL '05: Proceedings of the 43rd*

Annual Meeting on Association for Computational Linguistics, pages 419–426, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[109] Guangyu Zhu, Timothy J. Bethea, and Vikas Krishna. Extracting relevant named entities for automated expense reimbursement. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 1004–1012, New York, NY, USA, 2007. ACM.

Appendix A

A Sample Tagged News Article

<Article id="156">

<ArticleText>
Ankara'da bombalı saldırı: 1 ölü

20 Mayıs 2003 tarihinde Ankara Kızılay'da bir kafede intihar saldırısı hazırlığı içindeyken bombanın yanlışlıkla patlaması üzerine ölen kişinin DHKP-C militanı Şengül Akkurt olduğu açıklandı. Sabah 09.15 sıralarında meydana gelen patlama sonucu, kafenin tüm camları kırıldı, 3. katın yan duvarı çöktü. Emniyet yetkilileri, kafeye bir sırt çantasıyla gelen intihar eylemcisinin, tuvalette bombayı üzerine yerleştirirken yanlışlıkla patlattığını açıkladı. Üzerinden herhangi bir belge çıkmayan ve yüzü parçalanan kadın eylemcinin parmak izleri alındı ve tüm illere gönderildi. Malatya'da yapılan incelemeyle ölen militanın isminin Şengül Akkurt olduğu saptandı. Bu bilgilere göre Şengül Akkurt, 2000 yılından bu yana Malatya Devlet Güvenlik Mahkemesi tarafından aranıyordu.

Bu arada İçişleri Bakanı Abdülkadir Aksu, intihar eylemcisinin büyük bir olasılıkla güvenlik kuvvetlerine karşı eylem yapacağını söyledi. Tahrip gücü çok yüksek olan bombanın menşeini belirlemek için olay yerinden toplanan parçalar kriminal laboratuvarına gönderildi. Kafede meydana gelen bombalı eylemle ilgili üzerinde koyu renk tişört ve mavi kot bulunan bir kişi aranırken, Ankara polisi şüpheli paket konusunda alarma geçti. AŞTİ, Metro ve halkın alışveriş yaptığı kalabalık yerlerde güvenlik önlemlerini arttırdı.
</ArticleText>

<Tokens>
<Token id="1" startPos="0" endPos="9" stringValue="Ankara'da" />

<Token id="2" startPos="10" endPos="17" stringValue="bombalı" />
<Token id="3" startPos="18" endPos="25" stringValue="saldırı" />
<Token id="4" startPos="25" endPos="26" stringValue=":" />
<Token id="5" startPos="27" endPos="28" stringValue="1" />
<Token id="6" startPos="29" endPos="32" stringValue="ölü" />
<Token id="7" startPos="36" endPos="38" stringValue="20" />
<Token id="8" startPos="39" endPos="44" stringValue="Mayıs" />
<Token id="9" startPos="45" endPos="49" stringValue="2003" />
<Token id="10" startPos="52" endPos="61" stringValue="tarihinde" />
<Token id="11" startPos="62" endPos="68" stringValue="Ankara" />
<Token id="12" startPos="69" endPos="79" stringValue="Kızılay'da" />
<Token id="13" startPos="80" endPos="83" stringValue="bir" />
<Token id="14" startPos="84" endPos="90" stringValue="kafede" />
<Token id="15" startPos="91" endPos="98" stringValue="intihar" />
<Token id="16" startPos="99" endPos="108" stringValue="saldırısı" />
<Token id="17" startPos="109" endPos="118" stringValue="hazırlığı" />
<Token id="18" startPos="119" endPos="129" stringValue="içindeyken" />
<Token id="19" startPos="130" endPos="138" stringValue="bombanın" />
<Token id="20" startPos="139" endPos="150" stringValue="yanlışlıkla" />
<Token id="21" startPos="151" endPos="160" stringValue="patlaması" />
<Token id="22" startPos="161" endPos="168" stringValue="üzerine" />
<Token id="23" startPos="169" endPos="173" stringValue="ölen" />
<Token id="24" startPos="174" endPos="181" stringValue="kişinin" />
<Token id="25" startPos="182" endPos="186" stringValue="DHKP" />
<Token id="26" startPos="186" endPos="187" stringValue="-" />
<Token id="27" startPos="187" endPos="188" stringValue="C" />
<Token id="28" startPos="189" endPos="197" stringValue="militanı" />
<Token id="29" startPos="198" endPos="204" stringValue="Şengül" />
<Token id="30" startPos="205" endPos="211" stringValue="Akkurt" />
<Token id="31" startPos="212" endPos="218" stringValue="olduğu" />
<Token id="32" startPos="219" endPos="228" stringValue="açıklandı" />
<Token id="33" startPos="228" endPos="229" stringValue="." />
<Token id="34" startPos="230" endPos="235" stringValue="Sabah" />
<Token id="35" startPos="236" endPos="238" stringValue="09" />
<Token id="36" startPos="238" endPos="239" stringValue="." />
<Token id="37" startPos="239" endPos="241" stringValue="15" />
<Token id="38" startPos="242" endPos="253" stringValue="sıralarında" />
<Token id="39" startPos="254" endPos="261" stringValue="meydana" />
<Token id="40" startPos="262" endPos="267" stringValue="gelen" />
<Token id="41" startPos="268" endPos="275" stringValue="patlama" />

<Token id="42" startPos="276" endPos="282" stringValue="sonucu" />
<Token id="43" startPos="282" endPos="283" stringValue="," />
<Token id="44" startPos="284" endPos="291" stringValue="kafenin" />
<Token id="45" startPos="292" endPos="295" stringValue="tüm" />
<Token id="46" startPos="296" endPos="303" stringValue="camları" />
<Token id="47" startPos="304" endPos="311" stringValue="kırıldı" />
<Token id="48" startPos="311" endPos="312" stringValue="," />
<Token id="49" startPos="313" endPos="314" stringValue="3" />
<Token id="50" startPos="314" endPos="315" stringValue="." />
<Token id="51" startPos="316" endPos="321" stringValue="katın" />
<Token id="52" startPos="322" endPos="325" stringValue="yan" />
<Token id="53" startPos="326" endPos="332" stringValue="duvarı" />
<Token id="54" startPos="333" endPos="338" stringValue="çöktü" />
<Token id="55" startPos="338" endPos="339" stringValue="." />
<Token id="56" startPos="340" endPos="347" stringValue="Emniyet" />
<Token id="57" startPos="348" endPos="359" stringValue="yetkilileri" />
<Token id="58" startPos="359" endPos="360" stringValue="," />
<Token id="59" startPos="361" endPos="367" stringValue="kafeye" />
<Token id="60" startPos="368" endPos="371" stringValue="bir" />
<Token id="61" startPos="372" endPos="376" stringValue="sırt" />
<Token id="62" startPos="377" endPos="387" stringValue="çantasıyla" />
<Token id="63" startPos="388" endPos="393" stringValue="gelen" />
<Token id="64" startPos="394" endPos="401" stringValue="intihar" />
<Token id="65" startPos="402" endPos="414" stringValue="eylemcisinin" />
<Token id="66" startPos="414" endPos="415" stringValue="," />
<Token id="67" startPos="416" endPos="425" stringValue="tuvalette" />
<Token id="68" startPos="426" endPos="433" stringValue="bombayı" />
<Token id="69" startPos="434" endPos="441" stringValue="üzerine" />
<Token id="70" startPos="442" endPos="456" stringValue="yerleştirirken" />
<Token id="71" startPos="457" endPos="468" stringValue="yanlışlıkla" />
<Token id="72" startPos="469" endPos="481" stringValue="patlattığını" />
<Token id="73" startPos="482" endPos="490" stringValue="açıkladı" />
<Token id="74" startPos="490" endPos="491" stringValue="." />
<Token id="75" startPos="492" endPos="501" stringValue="Üzerinden" />
<Token id="76" startPos="502" endPos="510" stringValue="herhangi" />
<Token id="77" startPos="511" endPos="514" stringValue="bir" />
<Token id="78" startPos="515" endPos="520" stringValue="belge" />
<Token id="79" startPos="521" endPos="529" stringValue="çıkmayan" />
<Token id="80" startPos="530" endPos="532" stringValue="ve" />
<Token id="81" startPos="533" endPos="537" stringValue="yüzü" />

<Token id="82" startPos="538" endPos="548" stringValue="parçalanan" />
<Token id="83" startPos="549" endPos="554" stringValue="kadın" />
<Token id="84" startPos="555" endPos="565" stringValue="eylemcinin" />
<Token id="85" startPos="566" endPos="572" stringValue="parmak" />
<Token id="86" startPos="573" endPos="579" stringValue="izleri" />
<Token id="87" startPos="580" endPos="586" stringValue="alındı" />
<Token id="88" startPos="587" endPos="589" stringValue="ve" />
<Token id="89" startPos="590" endPos="593" stringValue="tüm" />
<Token id="90" startPos="594" endPos="600" stringValue="illere" />
<Token id="91" startPos="601" endPos="611" stringValue="gönderildi" />
<Token id="92" startPos="611" endPos="612" stringValue="." />
<Token id="93" startPos="613" endPos="623" stringValue="Malatya'da" />
<Token id="94" startPos="624" endPos="631" stringValue="yapılan" />
<Token id="95" startPos="632" endPos="643" stringValue="incelemeyle" />
<Token id="96" startPos="644" endPos="648" stringValue="ölen" />
<Token id="97" startPos="649" endPos="658" stringValue="militanın" />
<Token id="98" startPos="659" endPos="666" stringValue="isminin" />
<Token id="99" startPos="667" endPos="673" stringValue="Şengül" />
<Token id="100" startPos="674" endPos="680" stringValue="Akkurt" />
<Token id="101" startPos="681" endPos="687" stringValue="olduğu" />
<Token id="102" startPos="688" endPos="696" stringValue="saptandı" />
<Token id="103" startPos="696" endPos="697" stringValue="." />
<Token id="104" startPos="698" endPos="700" stringValue="Bu" />
<Token id="105" startPos="701" endPos="710" stringValue="bilgilere" />
<Token id="106" startPos="711" endPos="715" stringValue="göre" />
<Token id="107" startPos="716" endPos="722" stringValue="Şengül" />
<Token id="108" startPos="723" endPos="729" stringValue="Akkurt" />
<Token id="109" startPos="729" endPos="730" stringValue="," />
<Token id="110" startPos="731" endPos="735" stringValue="2000" />
<Token id="111" startPos="736" endPos="744" stringValue="yılından" />
<Token id="112" startPos="745" endPos="747" stringValue="bu" />
<Token id="113" startPos="748" endPos="752" stringValue="yana" />
<Token id="114" startPos="753" endPos="760" stringValue="Malatya" />
<Token id="115" startPos="761" endPos="767" stringValue="Devlet" />
<Token id="116" startPos="768" endPos="776" stringValue="Güvenlik" />
<Token id="117" startPos="777" endPos="786" stringValue="Mahkemesi" />
<Token id="118" startPos="787" endPos="797" stringValue="tarafından" />
<Token id="119" startPos="798" endPos="808" stringValue="aranıyordu" />
<Token id="120" startPos="808" endPos="809" stringValue="." />
<Token id="121" startPos="811" endPos="813" stringValue="Bu" />

<Token id="122" startPos="814" endPos="819" stringValue="arada" />
<Token id="123" startPos="820" endPos="828" stringValue="İçişleri" />
<Token id="124" startPos="829" endPos="835" stringValue="Bakanı" />
<Token id="125" startPos="836" endPos="846" stringValue="Abdülkadir" />
<Token id="126" startPos="847" endPos="851" stringValue="Aksu" />
<Token id="127" startPos="851" endPos="852" stringValue="," />
<Token id="128" startPos="853" endPos="860" stringValue="intihar" />
<Token id="129" startPos="861" endPos="873" stringValue="eylemcisinin" />
<Token id="130" startPos="874" endPos="879" stringValue="büyük" />
<Token id="131" startPos="880" endPos="883" stringValue="bir" />
<Token id="132" startPos="884" endPos="894" stringValue="olasılıkla" />
<Token id="133" startPos="895" endPos="903" stringValue="güvenlik" />
<Token id="134" startPos="904" endPos="916" stringValue="kuvvetlerine" />
<Token id="135" startPos="917" endPos="922" stringValue="karşı" />
<Token id="136" startPos="923" endPos="928" stringValue="eylem" />
<Token id="137" startPos="929" endPos="939" stringValue="yapacağını" />
<Token id="138" startPos="940" endPos="947" stringValue="söyledi" />
<Token id="139" startPos="947" endPos="948" stringValue="." />
<Token id="140" startPos="949" endPos="955" stringValue="Tahrip" />
<Token id="141" startPos="956" endPos="960" stringValue="gücü" />
<Token id="142" startPos="961" endPos="964" stringValue="çok" />
<Token id="143" startPos="965" endPos="971" stringValue="yüksek" />
<Token id="144" startPos="972" endPos="976" stringValue="olan" />
<Token id="145" startPos="977" endPos="985" stringValue="bombanın" />
<Token id="146" startPos="986" endPos="994" stringValue="menşeini" />
<Token id="147" startPos="995" endPos="1005" stringValue="belirlemek" />
<Token id="148" startPos="1006" endPos="1010" stringValue="için" />
<Token id="149" startPos="1011" endPos="1015" stringValue="olay" />
<Token id="150" startPos="1016" endPos="1024" stringValue="yerinden" />
<Token id="151" startPos="1025" endPos="1033" stringValue="toplanan" />
<Token id="152" startPos="1034" endPos="1042" stringValue="parçalar" />
<Token id="153" startPos="1043" endPos="1051" stringValue="kriminal" />
<Token id="154" startPos="1052" endPos="1066" stringValue="laboratuvarına" />
<Token id="155" startPos="1067" endPos="1077" stringValue="gönderildi" />
<Token id="156" startPos="1077" endPos="1078" stringValue="." />
<Token id="157" startPos="1079" endPos="1085" stringValue="Kafede" />
<Token id="158" startPos="1086" endPos="1093" stringValue="meydana" />
<Token id="159" startPos="1094" endPos="1099" stringValue="gelen" />
<Token id="160" startPos="1100" endPos="1107" stringValue="bombalı" />
<Token id="161" startPos="1108" endPos="1115" stringValue="eylemle" />

```
    <Token id="162" startPos="1116" endPos="1122" stringValue="ilgili" />
    <Token id="163" startPos="1123" endPos="1131" stringValue="üzerinde" />
    <Token id="164" startPos="1132" endPos="1136" stringValue="koyu" />
    <Token id="165" startPos="1137" endPos="1141" stringValue="renk" />
    <Token id="166" startPos="1142" endPos="1148" stringValue="tişört" />
    <Token id="167" startPos="1149" endPos="1151" stringValue="ve" />
    <Token id="168" startPos="1152" endPos="1156" stringValue="mavi" />
    <Token id="169" startPos="1157" endPos="1160" stringValue="kot" />
    <Token id="170" startPos="1161" endPos="1168" stringValue="bulunan" />
    <Token id="171" startPos="1169" endPos="1172" stringValue="bir" />
    <Token id="172" startPos="1173" endPos="1177" stringValue="kişi" />
    <Token id="173" startPos="1178" endPos="1187" stringValue="aranırken" />
    <Token id="174" startPos="1187" endPos="1188" stringValue="," />
    <Token id="175" startPos="1189" endPos="1195" stringValue="Ankara" />
    <Token id="176" startPos="1196" endPos="1202" stringValue="polisi" />
    <Token id="177" startPos="1203" endPos="1210" stringValue="şüpheli" />
    <Token id="178" startPos="1211" endPos="1216" stringValue="paket" />
    <Token id="179" startPos="1217" endPos="1226" stringValue="konusunda" />
    <Token id="180" startPos="1227" endPos="1233" stringValue="alarma" />
    <Token id="181" startPos="1234" endPos="1239" stringValue="geçti" />
    <Token id="182" startPos="1239" endPos="1240" stringValue="." />
    <Token id="183" startPos="1241" endPos="1245" stringValue="AŞTİ" />
    <Token id="184" startPos="1245" endPos="1246" stringValue="," />
    <Token id="185" startPos="1247" endPos="1252" stringValue="Metro" />
    <Token id="186" startPos="1253" endPos="1255" stringValue="ve" />
    <Token id="187" startPos="1256" endPos="1262" stringValue="halkın" />
    <Token id="188" startPos="1263" endPos="1272" stringValue="alışveriş" />
    <Token id="189" startPos="1273" endPos="1280" stringValue="yaptığı" />
    <Token id="190" startPos="1281" endPos="1290" stringValue="kalabalık" />
    <Token id="191" startPos="1291" endPos="1299" stringValue="yerlerde" />
    <Token id="192" startPos="1300" endPos="1308" stringValue="güvenlik" />
    <Token id="193" startPos="1309" endPos="1320" stringValue="önlemlerini" />
    <Token id="194" startPos="1321" endPos="1329" stringValue="arttırdı" />
    <Token id="195" startPos="1329" endPos="1330" stringValue="." />
</Tokens>

<Sentences>
    <Sentence id="1" startToken="1" endToken="6" />
    <Sentence id="2" startToken="7" endToken="33" />
    <Sentence id="3" startToken="34" endToken="55" />
```

```
    <Sentence id="4" startToken="56" endToken="74" />
    <Sentence id="5" startToken="75" endToken="92" />
    <Sentence id="6" startToken="93" endToken="103" />
    <Sentence id="7" startToken="104" endToken="120" />
    <Sentence id="8" startToken="121" endToken="139" />
    <Sentence id="9" startToken="140" endToken="156" />
    <Sentence id="10" startToken="157" endToken="182" />
    <Sentence id="11" startToken="183" endToken="195" />
  </Sentences>

  <Topics>
    <Topic id="1" startToken="1" endToken="195" />
  </Topics>

  <NEs>
    <Location id="1" startToken="1" endToken="1" stringValue="Ankara'da" />
    <Date id="2" startToken="7" endToken="9" stringValue="20 Mayıs 2003" />
    <Location id="3" startToken="11" endToken="11" stringValue="Ankara" />
    <Location id="4" startToken="12" endToken="12" stringValue="Kızılay'da" />
    <Organization id="5" startToken="25" endToken="27" stringValue="DHKP-C" />
    <Person id="6" startToken="29" endToken="30" stringValue="Şengül Akkurt" />
    <Time id="7" startToken="35" endToken="37" stringValue="09.15" />
    <Location id="8" startToken="93" endToken="93" stringValue="Malatya'da" />
    <Person id="9" startToken="99" endToken="100" stringValue="Şengül Akkurt" />
    <Person id="10" startToken="107" endToken="108" stringValue="Şengül Akkurt" />
    <Date id="11" startToken="110" endToken="110" stringValue="2000" />
    <Organization id="12" startToken="114" endToken="117" stringValue="Malatya De-
vlet Güvenlik Mahkemesi" />
    <Person id="13" startToken="125" endToken="126" stringValue="Abdülkadir Aksu"
/>
    <Location id="14" startToken="175" endToken="175" stringValue="Ankara" />
    <Organization id="15" startToken="183" endToken="183" stringValue="AŞTİ" />
  </NEs>

  <Relations>
    <LocatedIn id="1" entityId="4" refId="3" inferredFromText="Ankara Kızılay'da" />
      <InferredFromToken id="1" tokenRef="11" text="Ankara" />
      <InferredFromToken id="2" tokenRef="12" text="Kızılay'da" />
    </LocatedIn>
    <AffiliatedWith id="2" entityId="6" refId="5" inferredFromText="DHKP - C militanı
```

Şengül Akkurt" />
<InferredFromToken id="1" tokenRef="25" text="DHKP" />
<InferredFromToken id="2" tokenRef="26" text="-" />
<InferredFromToken id="3" tokenRef="27" text="C" />
<InferredFromToken id="4" tokenRef="28" text="militanı" />
<InferredFromToken id="5" tokenRef="29" text="Şengül" />
<InferredFromToken id="6" tokenRef="30" text="Akkurt" />
</AffiliatedWith>
</Relations>

</Article>

Appendix B

Named Entity Classes

- PERSON
- LOCATION
- ORGANIZATION
- DATE
- TIME

Appendix C

Entity Relation Classes

- LOCATED_IN
- AFFILIATED_WITH
- ATTACKED_BY

Appendix D

List of the used Gazetter Lists

- ORGANIZATION

  List of Common Organization Names

  List of Common Words appear in the first position of Organization Names

  List of Common Words appear in the last position of Organization Names

  List of Common Words preceding Organization Names

  List of Common Words succeeding Organization Names

  List of Common Words appear in Organization Names

- LOCATION

  List of Cities

  List of Continents

  List of Countries

List of Districts

List of Mountains

List of Geographical Regions

List of Rivers

List of Seas

List of Villages

List of Common Words preceding Location Names

List of Common Words succeeding Location Names

List of Common Words appear in Location Names

- PERSON

  List of Common First Names

  List of Common Last Names

  List of Common Person Titles (Long)

  List of Common Person Titles (Short)

  List of Common Words preceding Person Names

  List of Common Words succeeding Person Names

  List of Common Words appear in Person Names

- DATE

  List of Days (Numbers)

  List of Days (Words)

  List of Months (Numbers)

  List of Months (Words)

  List of Year Expressions (Numbers)

  List of Common Words preceding Date Expressions

  List of Common Words succeeding Date Expressions

  List of Common Words appear in Date Expressions

- TIME

  List of Hour Expressions (Numbers)

  List of Minute Expressions (Numbers)

  List of Common Words appear in Time Expressions

- NUMBER

  List of Numbers (Digits)

  List of Numbers (Words)

- GENERAL

  List of Stop Words