A HYBRID METHOD FOR TOPONYM RECOGNITION ON INFORMAL TURKISH TEXT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MERYEM SAĞCAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

AUGUST 2014

Approval of the thesis:

A HYBRID METHOD FOR TOPONYM RECOGNITION ON INFORMAL TURKISH TEXT

submitted by **MERYEM SAĞCAN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _____

Assoc. Prof. Dr. Pınar Karagöz
Supervisor, **Computer Engineering Department, METU** _____

Examining Committee Members:

Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Pınar Karagöz
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Osman Abul
Computer Engineering Department, TOBB UET _____

Assoc. Prof. Dr. Tolga Can
Computer Engineering Department, METU _____

Dr. Ruket Çakıcı
Computer Engineering Department, METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MERYEM SAĞCAN

Signature :

ABSTRACT

A HYBRID METHOD FOR TOPONYM RECOGNITION ON INFORMAL TURKISH TEXT

Sağcan, Meryem

M.S., Department of Computer Engineering

Supervisor : Assoc. Prof. Dr. Pınar Karagöz

August 2014, 86 pages

Since accessing the Internet is getting easier and people are more willing to share information on the Internet than the previous generations, the data on such kind of reachable sources are growing very rapidly day by day. Moreover, because of the popularity and widely usage of those sources, the information which researchers and organizations are interested in can be found somewhere in these data collection. The purpose of Information Extraction (IE) is to analyze this information cloud and to extract the desired data among them. This study designs a system dealing with a subfield of Information Extraction, namely, Named Entity Recognition (NER), which many of the IE systems use as a basis.

NER is used to identify the entities related to the aspired information in texts and classify them into a set of predefined categories such as person, location, and organization names, date and money expressions, etc. Since most of the desired information such as trends, agendas, needs and thoughts of people may vary among locations and a location name can be used for more than one location, extracting location information is another research area. There is a field for this purpose, named as Toponym Extraction, which uses NER as a basic step in order to recognize location names.

Toponym Extraction consists of two steps, namely Toponym Recognition and Toponym Resolution. The first step, Toponym Recognition, is the subject of the pro-

posed study. It aims to extract named entities referring to location names; whereas, Toponym Resolution aims to make decision about which geographical coordinate the entity refers to; since, a location name can be used for more than one geographical coordinates.

Prominence of social media such as Twitter and Facebook have drawn attention from companies and researchers interested in detecting trends; however, the informal and popular nature of these services leads to a large amount of noisy misspellings, lack of punctuation, non-standard abbreviations and abnormal capitalization which make the recognition process really hard. This case creates a new challenge in NER field; thus, it also creates a new challenge in Toponym Recognition.

The proposed system in this thesis, constructs a hybrid NER system which uses both rule based and machine learning based techniques to extract toponyms from an informally written, unstructured text document which includes Turkish tweets. In this study, Conditional Random Fields (CRF) is used as a machine learning tool and some features such as POS-Tags and Conjunction Window are defined to train the constructed CRF model. In the rule based part, regular expressions which aim to define some rules in order to extract some words that containing "köy", "deniz", "şehir", "istan", etc. are used. The result of the rule based part is used as a feature in the machine learning part. All defined features are experimented interchangeably and incrementally. In addition, various learning mechanisms within CRF are compared in terms of their accuracy. Finally, the proposed study shows the effect of the size of the training and test data sets on the system accuracy. Those parameters are all experimented and the combination giving the best result is used in the comparison part in which the system is compared with some previous studies.

Keywords: Named Entity Recognition, Twitter, Conditional Random Fields, Information Extraction, Toponym Recognition, Turkish Named Entity Extraction

ÖZ

GÜNDELİK TÜRKÇE METİNLERDE HİBRİT YÖNTEMLE YER İSİMLERİNİ TANIMA

Sağcan, Meryem
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Pınar Karagöz

Ağustos 2014 , 86 sayfa

İnternet erişiminin ve internette bilgi paylaşmaktan çekinmeyen kişi sayısının artmasıyla beraber bu tarz kaynaklardaki bilgi birikimi hızla çoğalmaktadır. İnsanlar kendileri, istekleri ve şikayetleri hakkında bilgi paylaştıkça büyüyen bu bilgi karmaşası içerisinden ihtiyaç duyulan verinin çıkarılabilmesi Bilgi Çıkarımı (BÇ) bilim dalının ilgi alanıdır. Bu tezde sunulan çalışmada BÇ bilim dalının alt dalı olan Varlık İsimlerini Tanıma (VİT) yöntemi kullanılmıştır.

VİT yöntemi, dokümanlardaki ulaşılmaya çalışılan bilgilere ait varlık isimlerini bulur ve bu bulguları önceden tanımlanmış kategorilere göre sınıflandırır. Bu kategoriler insan isimleri, yer isimleri, tarih ve para miktarı gibi ifadelerdir. Moda, gündem, insanların düşünce ve ihtiyaçları değişik yerler arasında büyük farklılıklar gösterdiği için ve bir yer ismi birden fazla koordinatı temsil edebileceği için, yer bilgisi çıkarma işlemi ayrı bir alan altında incelenmektedir. Yer İsimleri Çıkarımı alanı bu amaç doğrultusunda çıkmıştır. İlk aşamada yer isimlerini tanıyabilmek için VİT yöntemlerini kullanmaktadır.

Yer İsimleri Çıkarımı işlemi iki kısımdan oluşmaktadır. İlk aşama Yer İsimleri Tanıma, ikinci aşama ise Yer İsimleri Çözümleme işlemleridir. Yer İsimleri Tanıma işlemi dokümanlardan yer ismi belirten varlık isimlerini çıkarmayı amaçlar. İkinci aşama ise, ilk kısımda bulunan yer isimlerinin gerçekte hangi coğrafik bölgeyi kas-

tettiğini bulmaya çalışır. Çünkü yeryüzünde aynı ismi taşıyan birden fazla coğrafik koordinat bulunabilir. Yer İsimleri Çıkarımı işleminin ilk aşaması olan Yer İsimleri Tanıma işlemi VİT alanının bir alt alanıdır.

İnsanların sosyal medyada oldukça aktif olmaları, Facebook ve Twitter gibi sosyal medyanın önde gelen temsilcilerinin, toplumun eğilimini bulmaya çalışan firma ve araştırmacıların dikkatlerini çekmelerine sebep olmuştur. Fakat bu tarz sosyal medyadan edinilen veri yapısal bozukluk, noklama işaretleri eksikliği, yanlış yazılmış kısaltmalar ve kurala uygun olmayan büyük harf kullanımları gibi fazlaca bozukluk içermektedir. Bu bozukluklar kelimelerin ve cümlelerin dolayısıyla da yer isimlerinin anlaşılmasını zorlaştırmaktadır.

Bu tezde sunulan sistem, kural tabanlı ve makine öğrenimi tabanlı iki yöntemi birleştirerek Türkçe tweet içeren dokümanlardan yer ismi belirten varlık isimlerini tanımaya çalışan hibrit bir sistemdir. Bu sistemde makine öğrenimi için Şartlı Rastgele Alanlar modeli kullanılmıştır. Bu modeli eğitebilmek için yer isimlerini tanımlayan özellikler (Sözcük Türü, Düzenli İfadelerden çıkarılan özellikler, Konjonksiyon Penceresi, vs.) kullanılmaktadır. Kural tabanlı kısım içinse, içerisinde geçtiği ya da arkasından geldiği kelimelerin yer ismi olma ihtimallerini artıran bazı karakter dizileri ("köy", "deniz", "şehir", "istan", vb.) için düzenli ifadeler tanımlanmıştır. Kural tabanlı aşamanın çıktıları makine öğrenimine dayalı model için girdi olarak kullanılmıştır.

Çalışma kapsamında bu özelliklerden bazı Özellik Takımları oluşturulup bunların her biri için bir test koşturulmuştur. İkinci çesit deneyde ise Şartlı Rastgele Alanları eğitmek için kullanılan farklı eğiticiler test edilmiştir. Son olarak eğitim ve test dosyalarının boyutları ve her birinin içindeki yer isimleri sayısı değiştirilerek, bu dosyaların sistem sonucu üzerindeki etkisi incelenmiştir. En iyi sonucu veren kombinasyon daha önceden gündelik Türkçe veri üzerine çalışan bazı sistemlerle karşılaştırmada kullanılacaktır.

Anahtar Kelimeler: Varlık İsmi Tanıma, Twitter, Şartlı Rastgele Alanlar, Bilgi Çıkarımı, Yer İsimleri Tanıma, Türkçe Varlık İsmi Tanıma

to my mom indeptedly
and
to my husband with love...

ACKNOWLEDGMENTS

I would like to express my deepest thanks to my supervisor Assoc. Prof. Dr. Pınar Karagöz for her encouragement and support throughout this study.

I would like to thanks Dilek Önal for her helping me at any stage of the thesis, especially gathering datasets.

Another special thanks go to Gülşen Eryiğit for her giving me a chance to use their pipeline, to Dilek Küçük for her support in experiments part of the thesis.

I would also like to thank to my mother-in-law, Reşide Kılınç, for her support and her insightful attitude to me.

I am deeply grateful to my mother, Cennet Susam, for her love and support. Without her, I could never been at this point.

Finally, very special thanks to the love of my life, my dear husband, Ali Kılınç, for his support, his believing in me even if I would lost belief in myself, and finally his giving me unlimited happiness, pleasure and love.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

IE	Information Extraction
BÇ	Bilgi Çıkarımı
NER	Named Entity Recognition
VİT	Varlık İsmi Tanıma
MALLET	MAchine Learning for LanguagE Toolkit
CRF	Conditional Random Fields
NLP	Natural Language Processing
ML	Machine Learning
POS-Tags	Part-Of-Speech Tags
HMMs	Hidden Markov Models
MEMMs	Maximum Entropy Markov Models

CHAPTER 1

INTRODUCTION

1.1 Information Extraction and Named Entity Recognition

Reaching to the desired information has always been a problem in the history since the data on reachable sources and the numbers of those resources are growing very rapidly day by day. The main reason is that accessing the Internet is getting easier and people are more willing to share information on the Internet than the previous generations. When the amount of information is getting huge, it is getting harder to find, access, analyze, and use valuable or desired information among those. The purpose of Information Extraction (IE) field is to analyze this information clutter and to extract the desired data among them.

IE aims to make some inference from unstructured data. More formally, it is dealing with the automatic extraction of structured information such as types of event, entities, and relationships between entities from unstructured sources. It results with much richer forms of queries on those messy and unstructured sources than searching with a keyword. The extraction of information from noisy, unstructured sources is a challenging task that a large number of researchers are working on for over two decades now [35].

Information extraction is used in many different domains such as machine learning, information retrieval, database, web and document analysis. In the document analysis domain, information extraction process is usually involved for language processing or linguistic decomposition to extract usable data from unformatted documents. One of the extraction tasks used in IE is concentrated around the identification of named

entities, such as people, company and location names and relationship among them from natural language text [35].

Named Entity Recognition (NER) is an important process for information extraction systems. NER process is used to identify the proper names in texts and classify them into a set of predefined categories such as person names, location names, organization names, date and time expressions etc. For NER task, a variety of techniques has been used [38]. These techniques can be grouped under three approaches:

- Rule Based (linguistic) approaches
- Machine Learning (ML) based approaches
- Hybrid Systems

In recent years, NER systems are quite evolved to extract information from formally written text documents. The new challenge is to achieve information extraction from informally written text such as e-mails, SMS messages, tweets, etc, since social media outlets such as Twitter and Facebook have drawn attention from companies and researchers interested in detecting trends. Since these sources can be also considered as news sources spreading like wildfire, processes on those information are crucial. The informal and popular nature of these services leads to a large amount of noisy misspellings, lack of punctuation, non-standard abbreviations and abnormal capitalization. Another problem with this information clutter is that it consists of short data because of the character limit of those services. These kind of disadvantages cause traditional Natural Language Processing (NLP) techniques to have lower accuracy than the accuracy obtained from processing of the structured text such as newswire articles [26], [22].

Toponyms are indispensable for researchers and organizations while detecting trends. Since the trend in one place can differ from another, it is crucial to extract toponyms initially for such purposes. However, the challenge which is mentioned for NER on informal data is also a problem in Toponym Recognition which applied on informally written documents.

The proposed system in this thesis constructs a hybrid NER system which uses both

rule based and machine learning based techniques to extract toponyms from an informally written, unstructured text document which includes Turkish tweets.

1.2 Motivation and Contributions

Recently, NER systems have changed their domain from analyzing formal texts to analyzing informal texts such as tweets, Facebook messages, SMS messages, etc. in order to obtain more accurate information about needs and thoughts of people. Location information in those sources is one of the key feature for such systems since trends, agendas, needs and thoughts of people in a place can be totally different from those in another place. Therefore, extracting location information which is known as Toponym Recognition from such sources is crucial. The main motivation in this study is that this domain is quite new for Turkish language; therefore, it needs to be paid attention because of the lack of a system which recognizes toponyms with high accuracy.

The proposed study is a NER task which uses both machine learning based and rule based approaches. The constructed system uses Conditional Random Fields (CRFs) model in machine learning part to create the graphical model of the unstructured data which is a plain text document includes Turkish tweets obtained from Twitter media.

CRF is chosen for machine learning part since it has some advantages over other models such as Maximum Entropy Markov Models (MEMMs) and Hidden Markov Models (HMMs). CRFs can handle multiple interacting features and long-range dependencies between tokens which frequently exist in real-life data whereas others require the independence of the tokens. In addition, it avoids the label bias problem which exists in other models since it is based on undirected graphical models. CRF overcomes both HMM and MEMM in real-life sequence labelling tasks.

The system needs some features in order to train CRFs. The first purpose is to analyze the impact of features on the accuracy of a NER system while the system is working with informal data. The feature combination used in the study, is different from other studies ([3], [30]) which also work on NER for Turkish language. To extract correct POS-Tags as far as possible, a normalizer is implemented like in [3]; however, it

makes different types of adjustments.

The proposed system is a hybrid NER system and it trains CRFs with some rule-based features as in [3]; however, instead of using huge gazetteers, it uses some regular expressions. Therefore, the system does not use a tagged test data set; since there are not any gazetteers to pre-check the test data set. The advantage of supporting machine learning-based approach with rule-based features is that if desired information can be expressed with some pre-defined rules, adding an extra feature to tokens obeying the rules makes their probability of being a location higher in machine learning part.

In this study, MALLET [25] is used as a tool to run CRFs algorithm. It offers different kinds of trainers which train CRFs in different ways (in Section 2.4). Effects of the approach of training CRFs are also analyzed.

The proposed system consists of different combinations in terms of features, CRFs training, and informal-to-formal conversion than the previous studies which apply NER on Turkish tweet data.

We can summarize the contibutions of this thesis work as following:

- Feature sets which are used in machine learning process differ from other NER systems which work on informally written Turkish texts. Some language independent features are also imported to the system to overcome the informality.

- For the machine learning model, CRFs is employed and with the help of MALLET, various training mechanisms are compared. Each training mechanism trains CRFs model with a different approach.

- The constructed system propose a normalization phase which deals with more simple but more common issues in the informally written Turkish texts. This kind of normalization yields less erroneous adjustments.

- Huge gazetteers are not used in the proposed system; therefore, there is no pre-checking phase before machine learning process. In order to use the advantage of combining a rule-based approach with a machine learning phase, some regular expressions are employed.

1.3 Thesis Organization

The next chapter reviews some studies related to general approach of NER systems which work on Turkish language, use informal data to extract information, and analyze informally written Turkish text. Third chapter gives basic information about tools and models which are used in the proposed system and Chapter 4 presents the proposed work in detail. In Chapter 5, approaches and the results of the experiments, and comparisons of the presented system with existing studies are given. The last chapter concludes the study with an overview and future works.

CHAPTER 2

BACKGROUND

In this chapter, a brief information about technics, tools and algorithms related to the proposed work are given. Firstly, toponym recognition is explained in Section 2.1 and the definition of morphologycal processing, some morphological features of Turkish language and a brief description about a morphological analysis tool, TRmorph, are given in Section 2.2. In Section 2.3, types of NER methods and CRFs which is a model used for machine-learning based NER, are represented. In the final section for this chapter, Section 2.4, a sequence labelling tool used in proposed work for tagging location entities with final classification labels is introduced.

2.1 Toponym Recognition

The purpose of Toponym Recognition field is to find references to geographic locations which is known as toponyms, in the text and then to assign each toponym to the correct geographic coordinates among the many possible interpretations. Extracting toponyms from textual content is named as *Toponym Recognition* where as interpreting the references is known as *Toponym Resolution*. These tasks have several difficulties. Understanding the natural language is a challenge for toponym recognition where as toponym resolution suffers from making a decision for finding which of the many possible locations is being referenced [21].

The first challenge is named as geo/non-geo ambiguity. Some location names can refer to another entities such as a person, an organisation, etc. For example, while **Paris** in **Paris Hilton** is a part of a named entity which refers to a person; **Paris** can

be a toponym referring to the city in France. Another challenge is geo/geo ambiguity. Since location names are not unique, there are more than one geographic coordinates which is named with the same named entity. For example, there are more than 30 locations which named as **Paris** on earth [20].

Toponym recognition can be considered as a sub-field of NER since toponym recognition is dealing with finding location entities in a textual content where as NER is dealing with finding entities of other types, such as names of people, organizations, date, etc, as well as location entities.

2.2 Morphological Analysis

Morphological analysis focuses on composition and organisation of words in human language and it has an important role in many computational linguistics applications. It aims to understand how simple, complex, and varied words are formed and at the end it generates a proper definition to a word. In general, morphological studies analyse the words in terms of their meaningful constituents, their composition rules from significant components, and their place in the sentences.

Morphological analysis is nominately important for some languages which is morphologically complex since in such languages, some of the linguistic information denoted by more than one word and the connections between the words are confined into only one word. It is means that morphology is not equally conspicuous in all spoken languages. For example, a language which uses one word to express an action uses more morphology than the language expressing the same action with more than one word. Thus, we can say that Turkish makes more use of morphology than English, since English do not use morphology to separate "I went", "You went" and "We went" (It uses two different word to express the action and the person who went) where as Turkish needs to investigate the action and the person making the action in only one word "Gittim", "Gittin", and "Gittik". More detailed information about morphology for different languages can be found in [12].

Since Turkish is an agglutinative language, morphologists have to deal with a complex morphology. Word generation processes in Turkish end up with words which

can be very long even sometimes correspond to an English sentence. For example, "İstanbullulaştıramadıklarımızdanmışsınız" is a sentence consisting of only one word in Turkish. Its English equivalent is "You are one of those who we could not convert to an İstanbulite" (Example is taken from [5]). Although this example is not used commonly in daily language, it explains the complex morphological structure of Turkish. In Turkish, mostly, a new word is generated by adding an affix at the end of a root just like the previous example; the process is named as Suffixation, and some suffixes attach to the stems recursively. Moreover, the vowel of a particular suffix varies according to the vowel harmony whereas some suffixes do not obey this rule (such as kalp-l**e**r, hakikat-s**i**z, etc...), some suffixes begin with "ç/c", "k/g", and "t/d" and the choice changes according to the last phonological unit in the stem (posta-**c**ı / süt-**ç**ü, etc), and while attaching a suffix to a root or a stem, if the last letter of the stem is vowel and the suffix starts with a vowel, then either the initial vowel of the suffix is deleted or a consonant (y, ş, s, or n) is inserted between. These are some morphological characteristics of Turkish language, for detail information [11] is a valuable source.

Most of the natural language applications; for example summarization, sequence tagging, machine translation, need basic language processes such as POS-tagging, tokenizing, chunking, etc. These needs make the morphological analysis process quite important. To meet the requirement, morphological analyzers are developed for various languages. Since each language has different morphological features, each of them has to have its own morphological analyzer. For Turkish, the language on which the proposed work applied, there are limited analyzers. Two of those are TRmorph [5], and CLARIN [27]. In this study, TRmorph is chosen for morphological analysis part.

2.2.1 TRmorph

TRmorph [5], which is an open-source finite-state morphological analyzer for Turkish, is used in this study. It is based on freely available software tools and resources, and it allows to modify the code in the direction of needs. It comes with different kind of tools for stemming/lemmatization, morphological segmentation, hyphenation, pre-

dicting unknown words. It can be customized for some common choices during the compilation such as whether it allows proper nouns not starting with a capital letter or not, whether tokens written with all capital letters are analysed or not, etc. Moreover, it allows to add or modify lexical entries. These capabilities make it a useful morphological analyzer.

TRmorph consists of two implementation; namely TRmorph Morphological Analyzer and TRmorph Disambiguator. TRmorph analyser gives more than one result for a word since a word can have multiple meanings or it can divide into its meaningful root and suffixes in multiple ways. The analyser does not use context information, i.e., it does not take the other words in the sentence into account.

Dismabiguator is responsible to choose the appropriate one from the results of the analyzer. It does its job by taking the context into account by considering the conditional probabilities of tag sequences. For example, a word can have more than one role in a sentence; in other words, a word can be a noun or an adjective in the sentence. ("Saçlarının rengi kızıl." —> noun / "Kızıl saçlarını dalgalandırdı" —> adjective). Morphological analyser gives multiple result for "kızıl". To be able to decide which tag is appropriate for this word, disambiguator must consider the other words in the sentence.

For more information about TRmorph, its webpage [6] can be visited.

2.3 NER Methods

In order to identify names from text, different methods are used. Those methods can be categorized into three groups, namely Rule-base NER, Machine Learning-base NER, and Hybrid NER.

Rule-based NER can be used, if it is possible to write a set of regular expressions that can extract the intended named entities. In other words, to be successful with rule-based NER, the natural language description and the rules employed to recognize the named entities using syntactic and lexical structure of the words need to be formulated. Besides the rules, some gazetteers or general dictionaries can be required

by the approach [18].

Machine learning-based NER systems uses machine learning methods. Machine learning methods are language independent; since, they do not analyze the meaning of the words if they are used as a feature. Even if machine learning methods are language independent, a feature can be dependent to the language such as POS-tags. However, there are available *Natural Language-Tools* (NL-Tools) for feature extraction to supply the need. Therefore, the development cycle of a machine learning system is generally faster than Rule-based development cycle.

The last group is Hybrid NER. A rule-based NER approach uses pre-defined rules (if some regular expressions can be defined for intended named entities) and machine learning-base NER approaches are widely used due to their ability to train easily, to adapt to different domains, etc. A hybrid NER system is a combination of both approaches. In other words, it combines a machine learning model, such as CRFs, Hidden Markov Model, etc. with some rule-based methods such as gazetteers, dictionaries, pre-defined rule sets, etc ([3], [43]). This work proposed a Hybrid NER. In the study, CRFs method is used for machine learning purpose and a set of some pre-defined rules are used which make the study Hybrid NER. Detailed information is presented in Chapter 4, Methodology.

2.3.1 CRFs as a Machine Learning based NER Method

CRFs [19] are undirected graphical models and a CRF is a type of discriminative probabilistic model. They generate a conditional probability distribution by using feature sets and they are used for the labelling and segmenting a structured data, such as sequential data. To label the sequential data such as natural language text, CRFs are needed to be trained with a training dataset which includes the same type of features with the sequential data to be tested. By adding new features, the impressive power of CRFs can be increased. In the NER processes, the input for CRF is a token sequence extracted from a sentence or a document and the state sequence is its corresponding label sequence.

In the case in which there are edges between the nodes of the graphical model, named

as Linear-chain CRFs, CRFs make assumption (first-order Markov assumption) and they are in the form of conditionally trained probabilistic finite state machines. [38].

Linear-chain CRFs are corresponding to finite automates and they can be seen approximately as conditionally-trained hidden Markov models (HMMs) or nearly a globally-normalized extension to Maximum Entropy Markov Models according to [24].

CRFs are defined in [24] as follows: For labeling and segmentation tasks, one of the most common methods is HMMs which identifies the most probable sequence of labels for any given token sequence. This model defines a joint probability distribution p(x, y) over observation sequences (x) and the corresponding label sequences (y). In order to define a joint distribution, HMMs enumerate all possible observation sequences. This process is difficult if the observation elements cannot be represented independently from the other elements in the observation sequence. However, observation sequences in the real-life have multiple interacting features and long-range dependencies between observation elements. CRFs defines a conditional probability over label sequences given a particular observation sequence x, rather than defining a joint distribution over both label and observation elements. They are used to label observation sequences, by choosing a label sequence yielding a maximum conditional probability. CRF has some advantages over HMMs. HMMs requires the independence of the observations whereas CRFs can handle multiple interacting features and long-range dependencies between observation elements which are frequently exist in real-life data. Another advantage of the CRF is that it avoids the label bias problem. This problem is exhibited by HMMs and MEMMs (Maximum Entropy Markov Models) since they are based on directed graphical models. CRF transcends both of these models in real-life sequence labeling tasks ([19], [32], [37]).

A CRF consists of a conditional distribution p(y|x) and an associated graphical structure.

In the modeling sequences in an undirected graphical structure, X is a set of input variables that are observed and Y is a set of output variables that will be predicted. As an example, the Figure 2.1 shows the graphical model of the linear-chain CRFs.

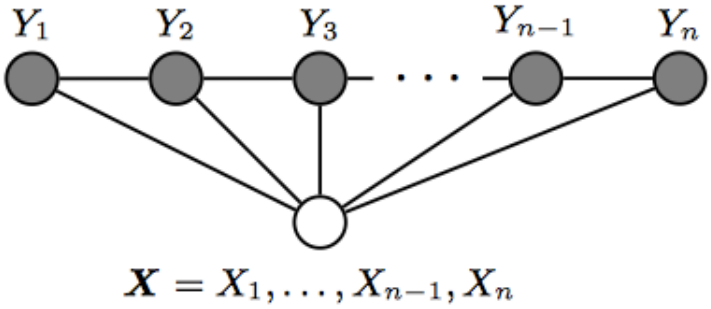

Figure 2.1: Graphical Model for Linear Chain CRF

The undirected graph, named as G, consists of vertices and edges, G = (V, E). Each vertex in V corresponds to an element in Y and if there is no edge between two of those vertices, it means that these vertices are conditionally independent. Therefore, the defined potential functions must ensure that conditionally independent random variables do not appear in the same potential function.

The conditional distribution of the linear-chain CRF is defined as p(y|x), and the probability of a particular label sequence y given observation sequence x is defined as:

$$P(\mathbf{y}, \mathbf{x}) = \frac{1}{Z_\theta(x)} \exp\left\{\sum_{t=1}^{T}\sum_{k=1}^{K} \theta_k f_k(y_{t-1}, y_t, x_t)\right\} \tag{2.1}$$

Whereas $f_k(y_{(t-1)}, y_t, x_t)$ is a potential function of transition from state y_{t-1} to state y_t with the input x_t and θ_k is the parameter optimized by the training. $Z_\theta(x)$ the normalization factor:

$$Z_\theta(x) = \sum_{y \in Y^T} \exp\left\{\sum_{t=1}^{T}\sum_{k=1}^{K} \theta_k f_k(y_{t-1}, y_t, x_t)\right\} \tag{2.2}$$

where y_t is the named entity label and x_t is feature vector containing all the components of the global observations x. [36]

A general CRF, used in information extraction, automatically build a relational database from information contained in the unstructured text. It can capture long-distance dependencies between labels unlike a linear-chain CRF and it is named as skip-chain

CRF. The skip-chain CRF has better performance than a linear-chain CRF [39].

In this proposed system, linear-chain CRF is used.

2.4 Mallet

Mallet consists of integrated java packages and it can be used for statistic NLP, information extraction, document classification, clustering, etc [25]. It comes with a wide variety of tools and implemented algorithms for some machine learning applications. For document classification Naive Bayes, Maximum Entropy, and Decision Tree algorithms are implemented. Mallet also implements sequence tagging algorithms to extract named entities from text. Those algorithms in Mallet for sequence tagging are HMMs, MEMMs, and CRFs. For topic modeling purpose, it has the sampling-based implementations of Latent Dirichlet Allocation, Pachinko Allocation, and Hierarchical LDA. To numerically optimize the algorithms Mallet has an efficient implementation of Limited Memory BFGS and some other numerical optimization methods. Lastly, MALLET has a "pipes" system to convert text documents into numerical representations. This process deal with some tasks such as conversion of sequences into count vector, tokenization, removing stopwords, etc. Detail information and MALLET Application Programming Interface (API) is available at [25].

The input data for Mallet is either a *Feature Vector* or a *Feature Sequence*. If the input is in the text format, MALLET can convert it into one of these formats with the help of its pipes. FeatureSequence is mutable and extendable storage for the object of the same class; where as, FeatureVector stores the words by assigning an integer to them; thus, all words are unique in FeatureVector.

Mallet uses different types of pipes in data importing and pre-processing phases. All classes implemented in Mallet and all pipes can be used with Mallet are described in Mallet's javadoc. Pipes which are employed in this study will be describe in the Proposed Work chapter.

In this study, Mallet is used with CRFs algorithm for sequence tagging. There are different types of methods to train CRF. In Mallet, implemented as CRF training mech-

anisms are CRFTrainerByL1LabelLikelihood, CRFTrainerByLabelLikelihood, CRFTrainerByStochasticGradient, CRFTrainerByThreadedLabelLikelihood and CRFTrainerByValueGradients.

CRFTrainerByLabelLikelihood, CRFTrainerByStochasticGradient, and CRFTrainerByValueGradients are used in experiments in this study and compared. The results are given in the Experiments and Results chapter.

2.4.1 Training CRF with Stochastic Gradient Trainer

Stochastic Gradient Descent (SGD) is an optimization algorithm and the basic idea of the SGD optimization is that it picks a training instance randomly and continuous slowly in the direction which is specified by the gradient for only the current instance. Stochastic Gradient trainer employes the log-likelihood objective function (more convenient than likelihood), which is the logarithm of the likelihood objective function [39]. In parallelization manner, it is not a good optimization method since the direction at which the steepest descent occurs can point a direction which is not the optimum. Directions are computed in label-likelihood or value gradient optimization (both of those use L-BFGS optimization) better than stochastic gradient optimization; whereas, stochastic gradient directions are computed much faster. However, the speedup which SGD optimization provides is only for large datasets; in other words, stochastic gradient optimization does not give a considerable performance on small datasets. Stochastic Gradient Optimization is going to be compared with other optimization methods in Chapter 5 in terms of their accuracy.

2.4.2 Training CRF with Label Likelihood Trainer

Label likelihood is an objective function for CRFs and its trainer tries to determine which label should be assigned to a given feature vector. Firstly, it calculates the probability for each target label by checking the occurrences of each label in the training data set. After calculating the frequency for each label, for each feature, it calculates the probability of being that feature with each target label together in the input training data set. To calculate the final probability, it multiplies all probabilities

of each feature in the feature vector of a token with a target label and it makes the same calculation for each label. Finally, it picks the label which gives the maximum probability as the estimated target label for that token [2]. The performance of the label likelihood trainer depends on the number of tokens in the training data set, the number of features, and the number of target labels. Therefore, label likelihood is not a good trainer for large training and test data sets or systems which have lots of features or have a large target label set. However, it can be easily parallelized since it does not decide the next direction by using the current instance.

2.4.3 Training CRF with Value Gradient Trainer

Value Gradient Trainer is a trainer for CRFs that can combine multiple objective functions, and each objective function is represented by an optimizable object. Optimizable objects can be optimizables by batch label-likelihood, by entropy regularisation, by value gradients and by label-likelihood. Those optimizables are objective function. In other words, value gradient trainer uses more than one objective function to make predictions on unlabeled data. The performance of the value gradient trainer changes according to the objective functions used as optimizables. For example, if batch label-likelihood is chosen as the objective function, the performance will be acceptable since it divides the input into multiple batches and then process each batches in parallel. For more information about learning with multiple objectives, [31] can be examined.

CHAPTER 3

LITERATURE REVIEW

The aims of NER systems is to extract and to classify named entities in a text document. NER tasks are used as a basic step for each IE system and some IE tasks such as event extraction performs NER task firstly.

There are three types of named entities, namely ENAMEX, TIMEX, and NUMEX. ENAMEX contains entities which refers to person, location and organization names. TIMEX type represents the time of day and date expressions. Entities referring to money and percent are analyzing under the NUMEX type [4].

Since real world domains are very interchangeable and very informal, learning based systems are getting more attractive because of their flexible nature. However, using some rules in the learning phase make the distinction between desired entities and others clear. Such kind of systems which uses both learning-based NER and rule-based NER are named as hybrid systems. The proposed work is a hybrid system which looking for toponyms in an unstructured informal data. There are various studies in NER field. Most of them proposed to work on formal data, such as news, whereas only few are designed for named entity extraction for informal data.

3.1 Named Entity Recognition Systems

Even though, the number of proposed studies is huge for NER field; many of those studies works on some specific languages, such as English. NER in some other languages on which several studies are proposed needs still paying attention because of

the lack of a system giving high accuracy. Since Turkish is one of those languages, analyzing studies which deals with a language which is not well-studied can give some clues.

There are lots of studies which apply NER on English text data by using different kind of features and techniques. For example; in [10], they combine classifiers which are maximum entropy, HMM, transformation-based learning, and the robust risk minimization classifiers and uses some features to identify named entities including words and their lemmas in a window which include 5 words at a time (surrounding the current word), POS-tags, text chunks in a [-1,+1] window, affixes, feature flags for each token such as allCaps, firstCap, 2digit, etc., and gazetteer information. The system has f-measure values of 91.15% for locations, 93.85% for person named and 84.67% for organization names. [14] is another example of NER on English texts. It uses CRF model and and some local features to make predictions. Features can be listed as a window which contains previous, current and next words, character n-grams and the shallow parse chunk of the current word, POS-tags of the tokens in the window and the presence of a word in a left window of size 5 around the current word. After CRF is trained with those local features, another CRF is employed by using the output of the first CRF and some majority features. Majority features can be exemplified as token-majority features (if there are two tokens which are both "Australia" and tagged as "Location", and another token which is also "Australia" but tagged as "Organization", the third token labeled as organization is tagged as location), entity-majority features, etc. By using all features, the system gives a f-measure of 87.24% for all entity types such as person, location, and organization.

[1] can be another example for NER systems but it does not work on a well-studied language. Dr. Balabantaray, Suprava Das and Tanaya Mishra have conducted a case study to exploit the conventional method for NER in Odia (an Indian language). They used part-of-speech (POS) tagger and gazetteers to generate feature sets for training and test data sets. Since English and Indian languages have structural differences, large number of POS-taggers can not be used for Odia. In this case study, a CRF-based POS-tagger which does not have satisfactory accuracy in general is implemented. Even though its performance is not acceptable, its accuracy of tagging proper nouns is quiet high. After tagging training and test data with POS-tagger, they

generate 3 different training and test data sets. The first column of the training data set is the word itself for all cases, the second column is POS-tags of the words for two training data sets where as it is only one of YES or NO tags (by using gazetteer) for the third file and the last column consists of named entity annotations generated by users. Test data sets involve words in the first column for all cases and POS-tags in the second column for two test data set and YES/NO tags for the last test file as it were in training data sets. Test data sets does not contain user generated tags. After the preparation of the system, they conducted a set of experiments. The performance is satisfactory for individual case; however, combining POS-tags and gazetteer tags decreases the performance of the system.

Another study is applied to explore features for NER in formal Lithuanian texts [13]. The aim is to solve a NER task for Lithuanian language which is not well-studied for NER tasks, and to explore different sets of features consisting of both language dependent and language independent features. Authors claim that in spite of high number of successful NER methods for English, for languages such as Lithuanian which is completely different from English in terms of grammatical structure, there is not any substantial study especially using supervised machine learning technics. In this research, PERSON, ORGANIZATION, and LOCATION tags are used to classify the named entities and CRF is used for the machine learning technic. Actually, there is 7 different tag for result since BIO tagging is used (B-LOC, I-LOC, B-PER, I-PER, B-ORG, I-ORG, O). The first step is tokenization and feature extraction. Each word is taken as a token and since in Lithuanian language, punctuation marks have a lot of information about named entities, they are also treated as a token. After tokenization step, some language dependent and independent features which are critical for supervised NER are extracted from dataset. Language independent features are basically token itself or punctuation mark, Lowercase (a function returns 1 if all letters in the word is lowercase, zero otherwise), isFirstUpper (whether the first letter is capitalised or not), Acronym (1 if all characters are capitalized, 0 otherwise), Number (whether the token is number or not), Length (length of the token), Prefix (determines the first n characters of the token (n is [3-5] in this study)), and Suffix (last n characters (n is [3-5] in this study)) features where as language dependent features are Lemma (transformation of the word into its major form), POS-tags, Stem (stem of

the token), IsPERGaz (whether the token in the PERSON gazetteer, or not), and IsLOCGaz (determines if the token in the LOCATION gazetteer). In addition, sliding window feature is also used. In experiments, 9 different sets of features are used with 12 different window size. Totally, 108 different experiments are conducted. Results of these experiments show that language dependent features increased the accuracy of NER and that affix information (prefix and suffix) achieved similar accuracy as with POS, stem and lemma features. The best window size which yields the best result is [-2, +1] and the highest F-Score is taken from the feature set in which all features are contained.

3.2 NER on Informal Texts

Since social media data has more up-to-date information than news, records, etc., it is critical to extract those informations to be able to track the trends, to analyze what people need, etc. However, it is a big challenge to recognize the entities which is not formally written. NER on those data is quite new and still needs contributions.

J. Lingad, S. Karimi and J. Yin analyze the applicability of NER for extraction of locations from microblogs [22]. The study can be a good example for the significancy of the information in the social media data. Location extraction from microblogs is critical since microblogs have valuable information which can impact a person life if a disaster is a matter of discussion. The purpose of the study is to be able to understand where the disaster affect, where help is needed and where this help is available. Location is defined as geographic location and point-of-interest by the authors. Geographic location means country, city, street, etc. and point-of-interest means hotels, pubs, etc. The objective of the research is to evaluate the effectiveness of major NLP tools with different settings and by using twitter data which are short and informal. *Stanford NER*, *OpenNLP*, *Yahoo! PlaceMaker*, and *TwitterNLP* are the tools used in the comparison. Experiments are performed with two settings: Out of the box (using already trained versions) and Retraining. Since Stanford NER and OpenNLP are trained for formal data, they are retrained for informal data. Another point which needs to be investigated is hashtags issue; because, they may indicate locations(exclude hashtags or include hashtags without hash symbol). According

to experiments of already trained versions, Stanford NER is giving the best result and Yahoo! PlaceMaker is in the second place with both excluding and including hashtags setting. TwitterNLP is worse than these two systems; even though, it is built for Twitter data. OpenNLP is the worst one. Experiments on retrained versions show that Stanford NER is still the winner. In general, these results are better than Out of the box results; because, tools are able to handle informality since they were retrained.

Nerit [8] is a technical report from The Johns Hopkins University and it proposes a language independent NER system. The system is also designed to work with short and informal data, such as twitter data. Since there is no specific language to work on, they do not use language dependent morphological analysis methods. For example, instead of using POS-tagging or gazetteers, more generic techniques are used to achieve some features to describe the word such as character composition of a word, the context of the word in a message, etc. All feature templates used in experiments are Token (word itself), N-gram (character n-gram of the word), Context (k words before and k words after), Length (message and word length), and Position (position of the word in the message). Token itself is baseline feature. Character n-gram is used to extract a word's prefix, suffix, or root; however, compared to morphological analysis, it provides a limited modelling of morphology and spelling error robustness. Another feature is Context (Sliding Window) and it identifies common word patterns around the current word to specify similar entities of that type. There are 10 types of entity in this system labeled as URL, Date, Time, Telephone, Person, Percent, Org (Organization), Money, Location and Email. The model used for training and testing phases is structural Support Vector Machine (SVM) which combines the discriminative learning of SVM with Hidden Markov Model. In experiments, both English and Spanish datasets are used and to analyse the each feature's contribution to model, each feature is evaluated independently.

The purpose of reviewing those kind of studies is to gain a perspective to be able to design a NER system working on Turkish data which is also written informally. Therefore, several number of studies are reviewed in addition to [8] and [22].

Xiaohua Liu, Shaodian Zhang, Furu Mei and Ming Zhou propose another hybrid NER

framework [43] which combines K-Nearest Neighbors (K-NN) classifier with linear CRF model to overcome informality and insufficiency of twitter data. First step is gathering the feature vector. After a word in a tweet is converted into a bag-of-words vector by KNN model (window size = 5), a feature matrix is created for that tweet. The created feature vector will be used in CRF model. Before running CRF model, KNN predicts the class of a word by leveraging the similar and recently labeled tweets and the class feature is also added to input of the linear CRF model. CRF conducts a tweet level NER. New labeled tweets are repeatedly added to the training set and when the number of new added training data exceeds a threshold number, the system (both CRF and KNN models) retrains itself. Before the system starts working, some improvements are conducted onto data such as removing stop words and normalizing the twitter meta data. Within the scope of this study, no additional features are used since to extract them requires a lot of computing resources. Similarly, since gazetteers contain noise, this also effects the performance negatively. In the scope of the study, some alternative models are also experimented, such as other classifiers based on Maximum Entropy and SVM instead of KNN and alternative NER models (Hidden Markov Model etc.) for CRF method. With this system, authors can labeled tweets using four types of label, namely LOCATION, PERSON, PRODUCT and ORGANIZATION. The overall experiment results are 81.6, 78.8, and 80.2 percent for precision, recall and F-Score respectively.

The final reviewed study is proposed by Allan Ritter, Sam Clark, Mausam and Oren Etzioni. They developed an experimental study, namely T-NER [34], in which they rebuild the NLP pipeline. In this study, they also propose a POS-tagger (T-POS), a shallow parser (T-CHUNK), and a capitalisation classifier (T-CAP). Since the state-of-the-art POS-taggers are not very useful for informal contexts, they build their own POS-tagger by using CRF. They annotated 16K tokens by using the Penn TreeBank tag set and new tags that they added for the twitter specific phenomena. Besides all these, for out-of-vocabulary words and lexical variations, hierarchical clustering (Jcluster) is performed. In addition to the clustering results, POS dictionaries, contextual features, spelling features, etc. are also used in CRF for T-POS. For T-CHUNK, the token set used in T-POS is annotated with tags from the CoNLL shared task. CRF is again used in shallow parsing step and for the features of the CRF, shallow pars-

ing feature set from a previous study described by Sha and Pereira [37], and Jcluster results are put in the training and test data sets. Detection of IOB encoded named entities is splitter into two step, namely, segmentation and classification of named entities and in the both part CRF is again used. In the segmentation of named entities step, orthographic, contextual, and dictionary features (set of type lists from Freebase), and the results of T-POS, T-CHUNK and T-CAP (which predicts that the tweet is informatively capitalized, or not) are employed as features. After segmentation, named entities needs to be classified. Since individual tweets do not contain enough information to classify entities, they take advantage of large lists of named entities and their types gathered from Freebase. However, some entities in tweets do not exist in Freebase baseline or they have more than one type. For this kind of unlabelled entities, LabelledLDA [33] is applied. Each entity is modelled as a mixture of types and associated with a bag-of-words within its context window in this method. It infers a distribution over possible types of the entity; thus, LabelledLDA solves non-availability of types and multiple types problem. After conducting experiments, obtained result is that T-NER doubles F-Score comparing against Stanford-NER.

3.3 Named Entity Recognition On Turkish Texts

NER in Turkish is quite well-studied on formal data. Since, the new trend in NER-world is to recognize entities in an informal data, there are several informal NER systems for Turkish and it is open for new perspectives.

3.3.1 NER On Formal Turkish Texts

The proposed system in [36] gives the highest accuracy among NER systems designed for formal Turkish data. The system, named as Turkish NLP Pipeline, makes some pre-processing on dataset to extract feature which will be given to the CRF. These pre-processes includes tokenization, morphological processing, and checking gazetteers steps. Firstly, the data is tokenized. In tokenization, proper nouns which is under inflection are separated into two tokens as the noun itself and the inflectional suffixes since they are separated by apostrophe. Each punctuation mark is treated as

a token. Each line has only one token and sentences are separated by an empty line in the output of the tokenizer. As the second step, morphological process is applied on tokenized data. In this step a two-level morphological analyzer, [27], is used. In the first level, the analyzer generates possible POS-tags for a token and in the second step, it chooses the most probable tag according to the context. After morphological processing, features which will be given to CRF are prepared by using the data coming from morphologycal processing and the gazetteers. They have two-types of gazetteers, namely base gazetteers and generator gazetteers. Base gazetteers includes all location names in Turkish postal code and telephone code system and for person names they also construct a base gazetteer from different sources. The second types of gazetteers are the generator gazetteers and they include some generator words, such as "bakanlık" (if it comes after some regular words such as spot, tarım, etc, it generates an organisation "Tarım Bakanlığı"). They use the situation of whether a token is in the base gazetteer or generator gazetteer or not. In addition to the morphological and gazetteers lookup features, they also use some lexical features, such as capitalization (the token in the proper name case, in the mixed case, in the all-uppercase or in the all-lowercase) and start-of-the-sentence (whether the token is at the first place in the sentence or not). In this study, a window is defined whose size is (-3, +3). By this window, features of the tokens which is in the window is also added as a feature for the current token. Finally, they take the results from CRF tool by using these features. The system gives the best result in the literature for NER on Turkish formal text. The f-measure in CONLL metric is 92.94% for person named entities, 88.77% for organisation names, 92.93% for location names and 91.94% for overall.

Another NER system which processing on formally written Turkish texts is proposed in [17]. This system employes a rule-based NER and it uses news texts, child stories, historical texts, and news video transcriptions as data in evaluation. The proposed system defines some rules to extract location, organization, and person names, time, date and money expressions; in addition to the some pre-defined lists. There are 4 types of lists in the study. Those lists are a dictionary containing person names in Turkish, a list of well known political people names in the world (such as Vladimir Putin, Ahmet Necdet Sezer, etc.), a list of well known location names in the world, and a list of well known organization names (such as Avrupa Birliği, Adalet Bakan-

lığı, etc.). In addition to these lists, 3 types of patterns are also defined. Those patterns are employed to extract location names (using patterns for "Sokak", "Yolu", "Kulesi", etc), organization names (using patterns for "Üniversitesi", "Partisi", "A.Ş.", etc.), and temporal and numeric expressions (using patterns for "X başı/ortası/sonu..." where X can be a 4-digit year name or the name of a month). By using these rules, authors evaluate their system with their own data set. The overall results of the evaluation is 75.8%, 81.8% and 78.7% for precision, recall and f-measure respectively.

3.3.2 NER On Informal Turkish Texts

Another study which constructs a NER system is [30], and this system analyzes informally written Turkish texts in order to extract desired information. Authors conducted their study on e-mail dataset which has informal context in the body part but has a well-defined header. In conducted study, a rule-based NER system is implemented by employing CRFs method. Each word in the e-mail is thought as a java object and all features which is used in CRF are defined as fields of the object. These fields are the word, word length, count, whether it is abbreviation or not, type (name, adverb, adjective, etc...), whether it is the first word of the sentence or not, etc. When the application runs, frequency of the word and character n-grams (n = 1, 2, 3) are extracted. Then, it is checked that the word is in the gazetteers (abbreviation gazetteers, title gazetteers, and special words gazetteers (Üniversite, Hanım, Bay, etc...)), that it starts with a capital letter, that it includes a punctuation mark, and that it is at the header of the e-mail. For each property, the corresponding field in the java object is set to 1. Finally, these properties are given to the CRF method to label the word. Besides these feature, they use (+1, -1) window size to be able to specify the named entities referring to name-surname sequence, n-grams features to identify the suffixes, and the length of the token to find the abbreviations. After extracting features, they takes the results of their system which gives 0.94, 0.97, 0.92 f-measures for academic, institutive, and personal e-mails respectively for person names where as for location names, f-scores are 0.68, 0.79 and 0.70 for the same types of e-mails.

[3] is a version of previously mentioned system, Turkish NLP Pipeline [36]. In this study, the proposed NER system is designed to work on informally written Turkish

text. This new system is experimented on twitter data, forum data and speech-to-text data. Since in the real data, there is lots of informality, the previous system needs to be adapted to the features of the new data. Firstly gazetteers are extended by adding some informal written names into the base gazetteers and some informal generators to the generator gazetteers such as "Abla", "Abi", "Hoca", etc. To make the morphologycal process give a result for the tokens which is written informally, a normalization phase is added to the system.The text normalizer works on the following cases:

- For slang words such as **"Nbr?"**. This word is adjusted as **"Ne haber?"**.
- For the tokens including repeated characters. For example: **"mutluyuuuum"**
- For hashtags, vocatives, mentions
- For emo-style writings such as **"$eker 4you"**
- For the tokens which is not capitalized correctly. For instance: **"istanbul"**

The rest of the system is same with the previously mentioned study. The system is experimented with different feature sets. CONLL metrics for feature set giving the best results are 91.64% for the news data with no normalization, 19.28% for twitter data with normalization, 50.84% for the speech data with normalization and 5.62% for the forum data.

[16] is the informal version of the study in [17]. In this study, the formal version ([17]) is used as a baseline, and its rules and resources are adapted to the new domain. The system is working on tweets and it needs to be adapted since the Twitter data contains grammatical errors, abnormal capitalization, contracted words instead of full forms, and words which is written by using English characters instead of the corresponding Turkish characters. In order to improve the performance of the initial system ([17]) on informally written tweets, capitalization constraint of the baseline system is turned off, and a phase which generates all possible tokens by interchanging the English-Turkish characters is added. To reduce the number of those possible tokens, they use a list of Turkish words which are written in their correct form and it checks whether a candidate is within this list. Besides those adaptations, a normalization phase is designed. In the normalization process, if a token includes consecutively repeated

characters and the token is not in the list which containing formally written words, then the repeated characters are contracted to one character. However, the system makes some wrong decision in normalization phase such as converting "Çanakkale" to "Çanakale". Since normalization process is applied before the NER system runs, a informal version of "şiir", "siir", is converted to "sir" which is also yields a wrong results. The evaluation of the system results with a overall f-measure which is 54.43% for person, location and organization types.

3.4 Toponym Recognition

Toponym Recognition is a part of the Toponym Extraction which consists of two steps. Another part of the field is Toponym Resolution (or Disambiguation). The proposed work in this thesis is a kind of Toponym Recognition system; however, it does not deal with the resolution of the toponyms. There are several studies on Toponym Extraction for English; whereas, for Turkish, there is not any system dealing with both recognition and resolution parts according to the best of our knowledge. However, [28] is a valuable study for Toponym Recognition.

In [20], Toponym Extraction is divided into two part, and the first part is recognizing the toponyms as usual. In order to recognize toponyms, they use Stanford-NER tool [9]; however, some features needs to be generated to be able to use this tool. To generate features, a POS-tagger is used to specify the proper nouns; since, locations tend to consist of proper nouns. They also use an entity dictionary which includes various types of entities such as location names, colours, seasons, religion, etc. to be able to recognize both toponyms and non-toponyms; since, it is important to know that a token is not a toponym in Toponym Resolution phase. After deciding toponyms according to the dictionary and extracting POS-tags of the tokens, Stanford-NER system is run with those features. When the NER system finishes tagging, there might be mistakes in selecting entity boundaries. In order to correct such mistakes, a post-processing step is designed. In post-processing step, boundary expansion method is used to correct the mistakenly selected entity boundaries. If an entity is found with a narrow boundary and the same entity is found with its correct boundary in one part of the document, the system expands its boundary according to the correctly found

entity. After extracting named entities referring to toponyms, Toponym Resolution process decide the geographical places to which the extracted entities refer.

In [38], toponym recognition is conducted with different kind of features from [20]. This system is also use POS-tags; however, it uses suffix&prefix, context words (features of the second previous, previous, next and second next tokens), some digit features and named entity tags of tokens within context words additionally. Digit features are extracted by analyzing tokens in terms of whether they contain digits or not and they consists of digits (2-digit or 4-digit) or not. In addition, if a token is a combination of digits and punctuation symbols, digits and periods, digits and symbols (Slash, Hyphen, etc), and digits and percentages, these features are also extracted and used as features for CRF. After extracting desired features, the system employes a CRF model by using these features. They use a named entity tagged geological corpus in the training phase. As a result of the experiments, the system gives precision, recall and f-measure metrics of 77.05%, 77.27%, and 75.81% respectively.

As it is mentioned before, according to best of our knowledge, there is not any system having both part of the Toponym Extraction process. However, [28] propose a Toponym Recognition system which works on Turkish Informal data. In the study, three types of Toponym Recognition approaches, namely gazetteer-based, rule-based and NER-based methods are experimented. Since the system working with informal data, a normalization phase is applied to the input data. While evaluating approaches, each one is experimented twice in the ways that normalization phase is included and excluded. In the gazetteer-based approach, three types of gazetteers (Cities of Turkey, Districts of Turkey and Turkish Names of Countries) are created. After tweets are morphologically processed and stem of the tokens are extracted, if a token is found in the one of the gazetteers, it is tagged as toponym. In rule-based approach, if a token ends with "e", "de", and "den" (which are suffixes representing dative, locative and ablative forms of the token respectively in Turkish), it is tagged as a location; since, in Turkish location names is usually used in one of those forms. In NER-based approach, pre-defined NER tools ([36] and [17]) are used as a named entity tagger. The first approach gives 43.19% f-measure value for normalized tweets as a result whereas the second one gives 28.37% f-measure for not normalized data and 28.03% f-measure for normalized data. Last experiment compares two previously developed

NER-based system and the resulted f-measures are 30.69% for [36] and 56.30% for [17].

3.5 Discussion

NER is a well-studied method to extract information from unstructured data. However, the most of produced systems depend on the languages. Actually, NER is well-studied for some languages such as English. The number of Turkish NER systems is much less than the number of those for English; however, there are some valuable Turkish NER systems which gives higher accuracy on formal written data. However, the new challenge is extracting desirable information from informal data; since the information on social media is valuable to understand needs of people, to previse the coming disasters, etc. The previous section reviews some studies which represent NER systems using hybrid NER model for different languages and for different types of data.

Almost all mentioned systems are using hybrid type of NER; however, they define different kind of features. Analyzing those studies gave an idea to design the proposed system.

POS-taggers are frequently used in NER systems even if they work on informal data. Besides POS-taggers, gazetteers are also preferred to pre-label the test data. Some of the systems working on informal data choose to implement its own POS-tagger ([34] is an example); since, existing taggers fails with informal data, and some of those defines a normalization phase to make adjustments on the data ([3]). The most important work among those studies is actually [3]. Since it works with informal twitter data which are written in Turkish, it is taken as a basis for the proposed work.

CHAPTER 4

METHODOLOGY

In this chapter, the proposed toponym recognition method is described in detail. The overall architecture is presented in Figure 4.1.

In order to use CRF implementation of MALLET, input data must be in an appropriate format. CRF needs a training data set where the location information is tagged correctly in order to extract toponyms from the given test data set. These two files are generated from a text based input file which contains Turkish tweets taken directly from Tweeter, by running the "Input Files Generation" part which is shown in Figure 4.1.

Formats of the training and the test data set is the same. The only difference is that toponyms are annotated in the training data set. Tokens of the data sets have to be analyzed in the same feature extraction processes. For example the morphological characteristics of each word in each tweet has to be determined in both data sets.

Tokens which do not contain location information are marked with "O". While the location information consists of a single token, it is tagged with only "B-LOC", if it includes multiple tokens, first token is tagged with "B-LOC" and the others are tagged with "I-LOC". For instance;

```
Bu Det:def O
akşam Adv O
Taksim N:prop B-LOC
Gezi N I-LOC
Parkı'ndayız N p3s loc 0 V cpl:pres 1p I-LOC
```

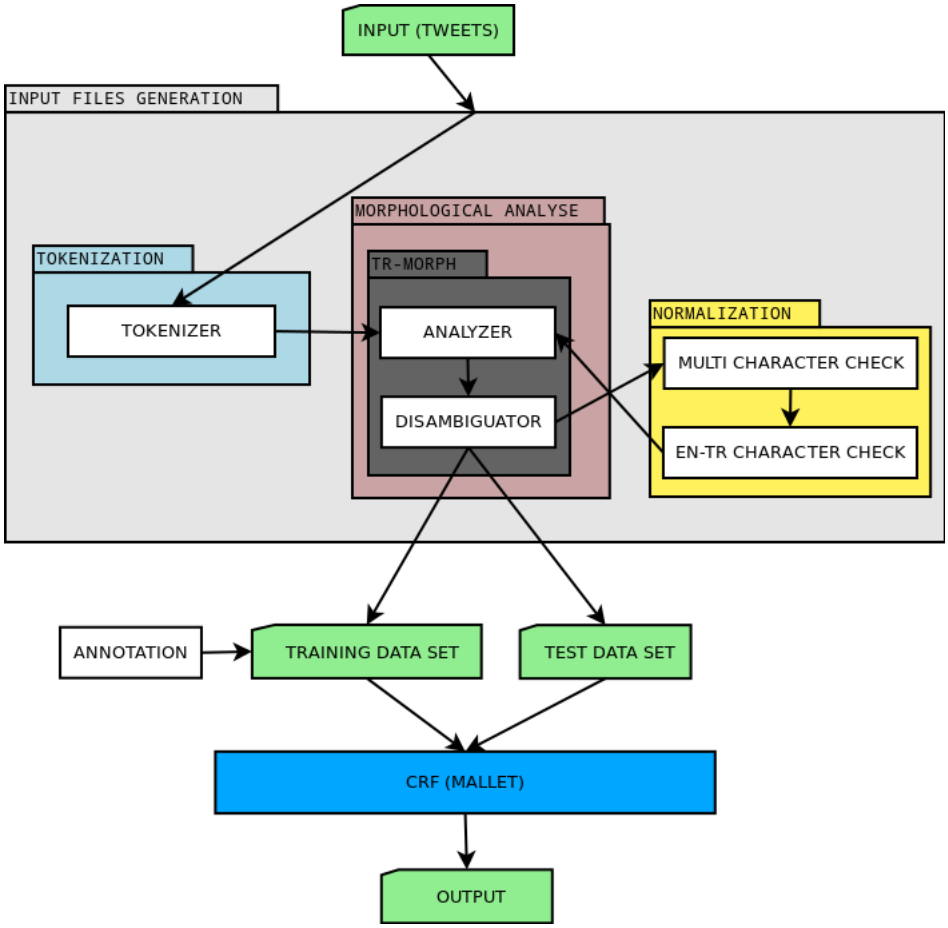

Figure 4.1: General Workflow of the Proposed Work

In the example, the tags such as “Det:def”, “Adv”, “N:prop” etc. are POS-tags which are the results of the morphological processing. CRF uses these tags by combining with other generated features in the stage of learning and extracting location information.

Consequently, tweets given in the input file are converted into CRF input files (test and training data sets) in the steps of Input Files Generation phase in Figure 4.1 and these files are used for training and testing in CRF phase with the help of MALLET.

4.1 Tokenization

Tokenization, in its simplest form, is the proccess of defragmentation of a text block into useful semantic units called tokens. These tokens are used in further processings such as feature extraction, training CRF model and testing. Basically, a token can be

defined as a word that are seperated by whitespaces, punctuation marks, line breaks etc. from its neighbour words in the sentence it belong.

The tokenizer that is used in this study, is a module which uses Java regular expressions in order to create tokens. The key features of this module;

- Splits a token sequence by using whitespaces and line breaks as a delimeter.
- Splits a token sequence according to punctuation marks that will not disrupt the integrity such as comma, dot if is not in the url, etc.
- Punctuation marks, whitespaces and line breaks are not included in the generated tokens list.
- Mentions and hashtags are not splitted and they are included in the token list.
- URLs are not splitted and they are also included in the final list.
- Tokens which includes apostrophe are not splitted, they accepted as one token. It is can be seen in the below example.

For the following example sentences, the tokenizer will give the results below:

“Halk Tv’de son dakika haberi; Beşiktaş, Galatasaray ve Fenerbahçe taraftarları otobüsle Taksim’e hareket ediyor.”

Halk
Tv’de
son
dakika
haberi
Beşiktaş
Galatasaray
ve
Fenerbahçe
taraftarları
otobüsle
Taksim’e

hareket
ediyor

"@istanbul Merter'de köprülü kavşağa 2 şerit eklendi karayolu üzerindeki #Merter ile İncirli http://t.co/RfoTBrLPFy."

@istanbul
Merter'de
köprülü
kavşağa
2
şerit
eklendi
karayolu
üzerindeki
#Merter
ile
İncirli
http://t.co/RfoTBrLPFy

As seen in the examples, each of those words seperated by an apostrophe ("Tv'de", "Taksim'e", "Merter'de") is treated as a single token because "de" and "e" suffixes changes the detail of the POS-tags of those tokens and also they are not meaningful as tokens themselves for TRmorph. Hashtags and mentions are not removed from the token and URLs are recognized as only one token.

The output of the tokenization process is a file containing single token on a line, and an empty line between sentences and it will be the input file for the Morphological Analysis phase.

4.2 Morphological Analysis

Morphological analysis phase takes the output of the tokenization process, and morphologically analyze each token. This phase is the where TRmorph morphological

analyzer tool is used.

The morphological analyzing phase consists of two steps:

- Morphological Analysis (TRmorph Morphological Analyzer)
- Morphological Disambiguation (TRmorph Morphological Disambiguator)

As a preparation for morphological analyze step, for each token, each letter in the token is capitalized and then each capitalized token is written per line in a file. This modification was needed since TRmorph works well with all-letter-capitalized words and it does not generate POS-tags for a token which does not start with a capital letter although it is a proper noun.

The second preparation is made in TRmorph implementations. Instead of giving each token itself to the analyzer or giving a group of the tokens, TRmorph analyzer is modified to be able to read the input file and analyze each token in the file.

After preparations, the pre-processed file is given to the TRmorph Morphological Analyzer. In the analysis phase, TRmorph processes each token individually. The analysis of each token can return multiple results since a word can have multiple meanings in Turkish, if it is not processed within a context. For example, while analyzing the sentence, "Kızıl gezegen hala sırlarla dolu.", for the word "kızıl", TRmorph gives the following result:

kızıl
kızıl kızıl<Adj>
kızıl kız<V><pass><V><imp><2s>
kızıl kızıl<Adj><0><V><cpl:pres><3p>
kızıl kızıl<Adj><0><N>
kızıl kızıl<Adj><0><N><0><V>
kızıl kızıl<Adj><0><N><0><V><cpl:pres><3p>
kızıl kızıl<Adj><0><N><0><V><cpl:pres><3s>
kızıl kızıl<Adj><0><V>
kızıl kızıl<Adj><0><V><cpl:pres><3s>

However, this word is certainly used in a way that maps only one of those morphological analyze result. This is why the morphological disambiguator is used after the morphological analysis. Morphological disambiguator takes the output of the analyzer and tries to find the correct analysis result by taking other tokens in the same sentence into consideration. For the word "kızıl", the disambiguator gives the following result by evaluating the token with the sentence to which it belongs; "Kızıl gezegen hala sırlarla dolu.":

kızıl kızıl<Adj>

Finally, after taking the disambiguator results, the output file is prepared. Each line of the file includes the word itself and its morphological results separated with a white space. Example output file for the sentence above is as follows:

kızıl Adj
gezegen N
hala N
sırlarla N pl ins
dolu Adj

A complete list of POS-tags that TRmorph generates can be found in the manual at [6].

Because of the informal nature of Twitter text, TRmorph does not have the ability to recognize such informally written tokens as a matter of course. Moreover, it may recognize some tokens wrongly, since the informally written form of a token can be the correct form of another word. For example, the result of the above sentence, if it was informal, would be as following;

kizil ???
gezegen gezegen<N>
hala hala<N> → WRONG! (Since it is mistyped, it means "aunt")
sirlarla ???
doluuuu ???

Since there are too much informality in the input files, a normalization phase is nec-

essary. Therefore, morphological analysis phase runs one more time in the Normalization part after informal to formal conversion; however, it is slightly different. This re-run step is explained in Section 4.3.

4.3 Normalization

Normalization phase generates possible formal forms of tokens which morphological analysis can not produce results for. At this stage two most common situation in the informal writing of Turkish is handled;

- Multi Character Check: Checking a certain letter which is written more than one time in a word.
- En-Tr Character Check: Checking interchangeably written Turkish – English characters.

Normalization phase gives all possible formal forms of an informal written word obtained after processing these two stages, to the Morphological Process part. The first possibility that Morphological Analyse produces result for, is accepted as the correct one and added to the main input file for CRF.

4.3.1 Multi Character Check

According to the morphological features of Turkish language, a letter can only be repeated two times at most, within a word. However, in Twitter, in order to emphasize some emotions or situations, some characters in a token can be used more than two times. This is one of the most encountered situation in social media data such as tweets, SMS messages, etc. Some example words for this kind of situation are given below:

```
Seviyoruuummm
Anneee
Hoşbulduuuk
```

`Geliyoooor`

Possible correct forms of these words are generated by using the repeating character one and two times in the word. All possible generations for the above examples after multi character check, are given in Table 4.1.

Table 4.1: Possible Words After Multi-Character Check

Seviyoruuuummm	Anneee	Hoşbulduuuk	Geliyoooor
Seviyorum	Ane	**Hoşbulduk**	**Geliyor**
Seviyorumm	**Anne**	Hoşbulduuk	Geliyoor
Seviyoruum	Anee		
Seviyoruumm	Annee		

All of these probabilities are given to the morphological analyse phase and 4 correct words (Seviyorum, Anne, Hoşbulduk, Geliyor) are found with POS-tags. However, after multi-character checking process, possible versions are not given to the morphological process directly. They are given as an input to English-Turkish character checking process and it also generates all possible versions for each of token generated by this step.

4.3.2 En – Tr Character Check

Another common informal writing habbit is using some Turkish – English characters interchangeably. Some of the letters in Turkish alphabet do not have equivalent English letters such as “ı”, “ğ”, “ö” etc. Instead of these letters, “i”, “g”, “o” are used.

In this stage of the normalization, possible correct forms are generated by interchanging these characters with its corresponding version. A complete list of these letters and character groups are given in Table 4.2.

If there are tokens which is output of the multi character checking stage, each of them is analyzed in this stage. If a token that is not recognized by TRmorph does not have multi-character inside, the original word itself is given to this phase. Example

Table 4.2: Interchangeable Characters List for English-Turkish Character Check

ç → c	Ç → C
c → ç	C → Ç
ı → i	I → İ
i → ı	İ → I
ğ → g	Ğ → G
g → ğ	G → Ğ
ö → o	Ö → O
o → ö	O → Ö
ş → s	Ş → S
s → ş	S → Ş
ü → u	Ü → U
u → ü	U → Ü
x → ks	X → KS
w → v	W → V
q → k	Q → K
sh → ş	SH → Ş
	€→ E
	$ → Ş

informal words are listed below. Figure 4.2 and Figure 4.3 show the possible words generation flow for two of those example words.

```
Aksam (instead of Akşam)
Taxi (instead of Taksi)
Kizil (instead of Kızıl)
Diger (instead of Diğer)
Sewiyorum (instead of Seviyorum)
Istanbuuul (instead of İstanbul)
```

Output of the multi character check stage is the same as the original word itself for

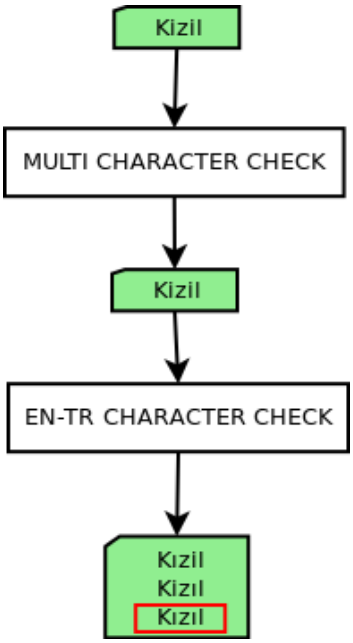

Figure 4.2: Example which does not have multi-character in the initial form

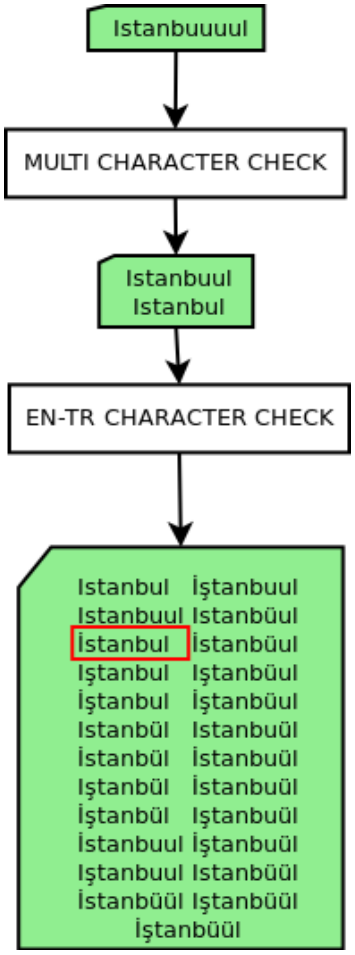

Figure 4.3: Example which have multi-character in the initial form

the token in Figure 4.2, since there are no repeating character in it. Output of the second stage generates three possibilities. As a result the Morphological Analyse phase produce pos tag(s) for only the word “Kızıl” which is the true form of the original word.

As shown in Figure 4.3, the output of the multi character check stage for the second example, “Istanbuuuul”, generates two possible words which contain the repeating character, “u”, one and two times. These two words are given to the En – Tr Character Checking stage as input and the output of this stage includes all possibilities according to the character conversions given in Table 4.2. At the end, the Morphological Analyse phase produce POS-tag(s) for only the word “İstanbul” which is the true form of the original word. In this Morphological Analyse phase, TRmorph chooses

the first possible form for which it can produce POS-tags as the true form of the token.

All processes in tokenization, morphological analysis and normalization phases could have been achieved with Zemberek-NLP tool. Zemberek can tokenize, normalize and then morphologically analyze a sentence. However, for normalization process, Zemberek gives a lot of suggestions and it did not convert some symbols such as "$", "€" to "Ş" and "E"; therefore, it is hard to select the correct form of an informal token and it generates wrong morphological features for tokens including "$" and "€" characters. Moreover, it is not capable of normalizing tokens which contain capital letters and punctuation marks in it. For tokenization task, Zemberek splits tokens which contain hashtags, mentions and most importantly, it splits URLs. Considering the disadvantages of Zemberek in normalization and tokenization processes, we implement our own tokenization and normalization phases. For morphological analysis, TRmorph is selected.

4.4 Extraction of Location Named Entities with CRFs

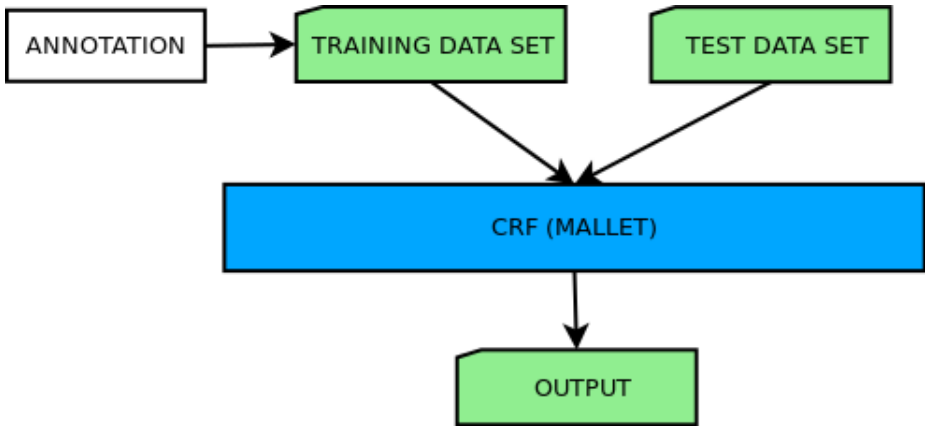

Figure 4.4: Training and Testing Phase

The final step in the workflow is to train a supervised learning model and testing the model. In previous phases, a training dataset is prepared; however, to take more accurate test results, more features need to be generated. In this phase, features are generated with appropriate MALLET pipes. The previously prepared training and test data sets include the token itself and its result from morphological analysis as features. These data sets are given to MALLET as input, and some new features

which will hopefully increase the accuracy at the end are added with MALLET pipes. All pipes and features are described below in the order of processing within Mallet and shown in Figure 4.5:

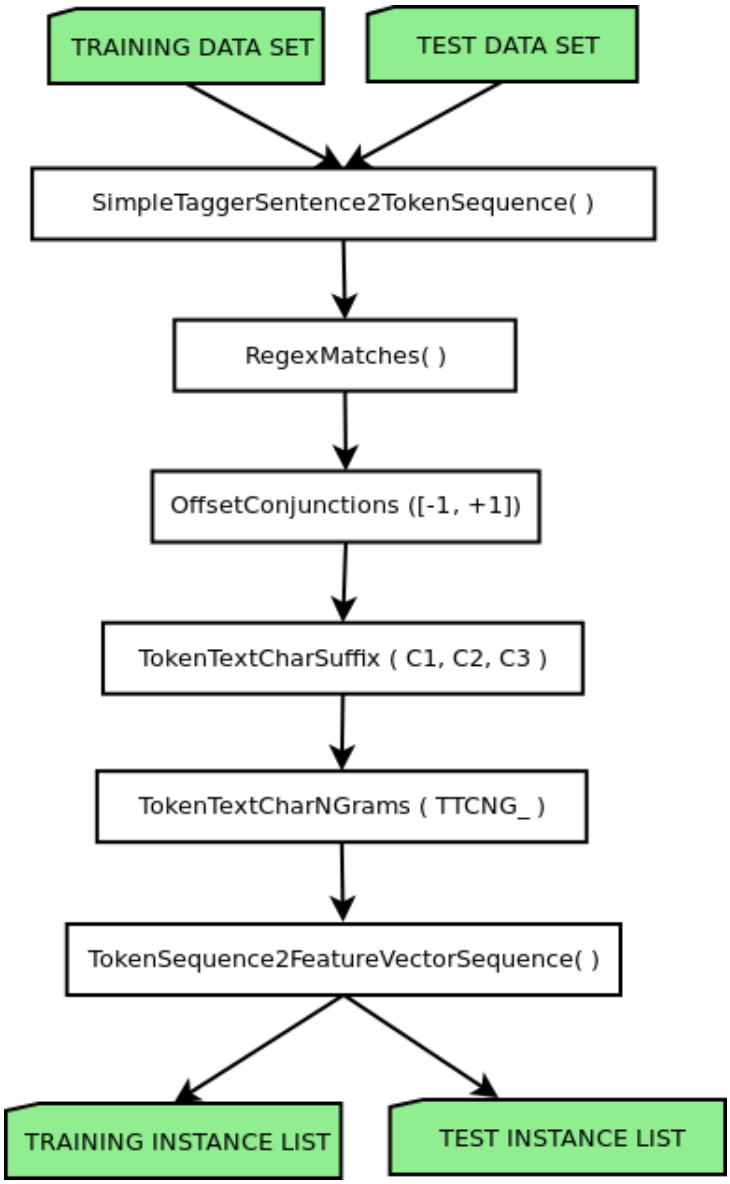

Figure 4.5: MALLET Pipes used in Proposed System

- **SimpleTaggerSentence2TokenSequence():**
  It converts a string consisting of lines of tokens which are whitespace-seperated or a two-dimensional string array into a TokenSquence which is a format that Mallet can process on it. Both structure, actually, represents row of tokens. This pipe also extracts the labels of the elements in training data set to create

a target label sequence. The last token in each row is selected the target label of the sequence in that line; where as, all other tokens in the same line are the features of the element represented by this row. In this study, 3 types of target label is used, namely B-LOC, I-LOC, and O.

B-LOC and I-LOC represents the tokens which belongs to a Location Named Entity; where as, O represents the token which is not a part of a named entity referring to a location. B and I stands for **B**eginning token of an entity, and **I**nner token of an entity, respectively.

- **RegexMatches(Tag, Regex):**
  This pipe is used to tag some tokens which corresponds to a regular expression pattern with some user-defined labels. Tag, the argument to the pipe, is the label and the other argument, Regex, is a pattern. If a token can be extracted with the Regex, then this token is labeled with the Tag. For example, one of the RegexMatches pipes used in the study is:

```
pipes.add(new RegexMatches("LOCATION",
            Pattern.compile("^.+[Dd]eniz.*")));
```

  This pipe adds "LOCATION" label as a feature for a token which satisfies the Pattern:

```
("^.+[Dd]eniz.*")
```

  The pattern extracts each word that contains "deniz" or "Deniz" in it, and the pipes add "LOCATION" feature to those tokens. However, the token which will be tagged as "LOCATION" can not start with "deniz" or "Deniz" since the purpose is to add feature to some tokens such as "Karadeniz", "Akdeniz", etc. Because, words including "deniz" at the beginning probably refers to a person, or refers to the word "sea".

  With the help of this pipe, tokens which contains deniz, şehir, köy, kuzey, güney, istan, etc. are featured as "LOCATION". However, this is not a final decision of course. The target tags in the study are O, B-LOC, and I-LOC. This

"LOCATION" label is just a feature which enhance the probability of the token being a location entity. These tokens are labelled; because, in Turkish, some words such as deniz, köy, şehir are frequently used in location names. "Karadeniz", "Kırşehir", "Mecidiyeköy" are some examples. Probably, tokens such as "cadde", "sokak" are also in a location named entity and it worths to tag these tokens as a "LOCATION" feature, too. Lastly, the tokens which includes "istan" in it are also tagged by this pipe; since, in Turkish some of country names ends with this suffix such as "Afganistan", "Hindistan", "Kazakistan", etc.

As a result, "LOCATION" feature tells CRF that there might be a location named entity around this feature.

- **OffsetConjunctions(conjunctions):**
  OffsetConjunctions takes a two dimensional integer array as an argument. For example:

```
int[][] conjunctions = new int[4][];
conjunctions[0] = new int[] { -1 };
conjunctions[1] = new int[] { 1 };
conjunctions[2] = new int[] { -2 };
conjunctions[3] = new int[] { 2 };
```

  According to this example, when a token is evaluated in this pipe, the pipe adds the features of previous token, next token, second previous token, and second next token to the feature list of the current token. An example syntax after this pipe executes for the token "dünya"; where as the sentence is "Bulutlar bu dünya için fazla güzeller":

  fazla@**2** Adj@**2** Bulutlar@**-2** N@**-2** pl@**-2** icin@**1** N:prop@**1** p2s@**1** bu@**-1** Det:def@**-1** Prn:pers:3p dünya

  In this notation, fazla@**2** shows that the second next token is "fazla", Adj@**2** shows that the second next token is an adjective, and N@**-2** reveals that the second previous token is a noun, etc.

- **TokenTextCharSuffix(Tag, Position):**
  This pipe is used to add suffix feature to the token. It is similar to Token-

TextCharPrefix which is used to add prefix features. Since Turkish is an agglutinating language, TokenTextCharPrefix is not employed in this study; where as, TokenTextCharSuffix used three times.

```
pipes.add(new TokenTextCharSuffix("C1=", 1));
pipes.add(new TokenTextCharSuffix("C2=", 2));
pipes.add(new TokenTextCharSuffix("C3=", 3));
```

These pipes add last character, last two and last three characters to the feature list of a token. For example, C3=dan C2=an C1=n are suffix feature for the token "birazdan". This feature is used to extract that the token is in form of dative, locative, ablative, etc.

- **TokenFirstPosition(Tag):**
  It adds Tag, the argument to the pipe, as a feature for the tokens in the first place of a sequence. A training or test data set consist of sequence of token sequences which are separated by an empty line. Each token at the first place of each token sequences has a feature named as "FIRSTTOKEN".

- **TokenTextCharNGrams(Tag, nGrams):**
  TokenTextCharNGrams takes out all n-consecutive characters from the token and each of those is added to the feature vector of that token as a feature. This pipe takes an integer array in which there are some **n**-values as the argument and then it extracts all n-consecutive characters for each of the values in the array. The argument, named as **Tag**, is used as the prefix for extracted features. For example, for the token "harikası", the pipe adds following features to the feature vector of this token when the nGrams array includes 3 and 7 for n-values and Tag is "TTCNG_".

```
int[] nGrams = new int[] {3, 7}
pipes.add(new TokenTextCharNGrams("TTCNG_", nGrams))
```

The Result for the token "harikası":

```
TTCNG_arikası TTCNG_harikas TTCNG_ası TTCNG_kas
TTCNG_ika TTCNG_rik TTCNG_ari TTCNG_har
```

- **PrintTokenSequenceFeatures(trainingFilename):**
  It prints all tokens gathering from training and test data set and all the features of each token which is already in the input data set and generated with Mallet pipes. In this workflow, this pipe is used just before the conversion from token sequence to feature vector sequence. This pipe has no effect on the CRF results. It just listing the tokens and their feature to make the process traceable. With the help of this pipe, the tokens and their features also including features generated by Mallet for the sentence "Beşiktaş diyorum iyiki var." can be seen below:

  - `TTCNG_eşiktaş TTCNG_Beşikta TTCNG_taş TTCNG_kta TTCNG_ikt TTCNG_şik TTCNG_eşi TTCNG_Beş FIRSTTOKEN C3=taş C2=aş C1=ş iyiki@2 <START1>@-2 diyorum@1 V@1 cont@1 1s@1 <START0>@-1 N:prop Beşiktaş`
  - `TTCNG_diyorum TTCNG_rum TTCNG_oru TTCNG_yor TTCNG_iyo TTCNG_diy C3=rum C2=um C1=m var@2 Exist@2 <START0>@-2 iyiki@1 Beşiktaş@-1 N:prop@-1 1s cont V diyorum`
  - `TTCNG_iki TTCNG_yik TTCNG_iyi C3=iki C2=ki C1=i .@2 Punc@2 Beşiktaş@-2 N:prop@-2 var@1 Exist@1 diyorum@-1 V@-1 cont@-1 1s@-1 iyiki`
  - `TTCNG_var C2=ar C1=r <END0>@2 diyorum@-2 V@-2 cont@-2 1s@-2 .@1 Punc@1 iyiki@-1 Exist var`
  - `<END1>@2 iyiki@-2 <END0>@1 var@-1 Exist@-1 Punc .`

- **TokenSequence2FeatureVectorSequence():**
  It converts all the feature sequence for each instance to a feature vector sequence which can be processed in CRF training part.

These pipes are added into a pipe arraylist and then a Pipe object is defined as SerialPipes which takes the pipe arraylist as an argument. SerialPipes makes Mallet

execute these pipes in the arralist serially. After that, an InstanceList object is created for the training data set and the other InstanceList is created for the test data set with this pipe. It means that both training data set and test data set are passed through the same pipe sequence; since, all tokens in the training and test data sets have to have same kind of features. However, Mallet does not oblige that each token will have the same number of feature. For example, a token has "LOCATION" feature; since, it corresponds one of the regular expressions; where as, another token does not have this feature; since, it does not match any one of those regular expressions. Therefore, the correct phrase should be that all tokens have to pass through the same analysis.

After preparing the test and training data sets, the next operation is training the CRF. Mallet has a CRF class and some kind of training methods. After creating the CRF object, a trainer is constructed which is a necessary for training and testing processes. CRF is trained with stochastic gradient, label likelihood, and value gradient trainers separately in this study.

CRF constructs a finite state machine [23] in order to model the unstructured data. Finite state machines consist of states which describe the status of the system. The system can be in only one state at a time. Each state in the constructed FSM is one of the feature label in the training token sequence and stop states are the target labels (B-LOC, I-LOC and O). After constructing the finite state machine, connection of states is defined. In this study, fully connected finite state machines are used. Connections are undirected and weighted. Weights are crucial for input feature/transition combinations. Weights are specified with setWeightDimensionsAsIn(trainingFile) method in which only parameters observed in the training data set are used for input feature/-transition combinations.

After defining the finite state machine properties such as its connection and its weights, finally a start state is added to the model with crf.addStartState() method of MALLET. These are general settings for all CRF trainers in MALLET and when these settings are done, a trainer for CRF can be selected. The workflow for training and testing process in MALLET is shown in 4.6

Firstly, a stochastic gradient trainer is selected for experiments in this study. The trainer definition is as the following:

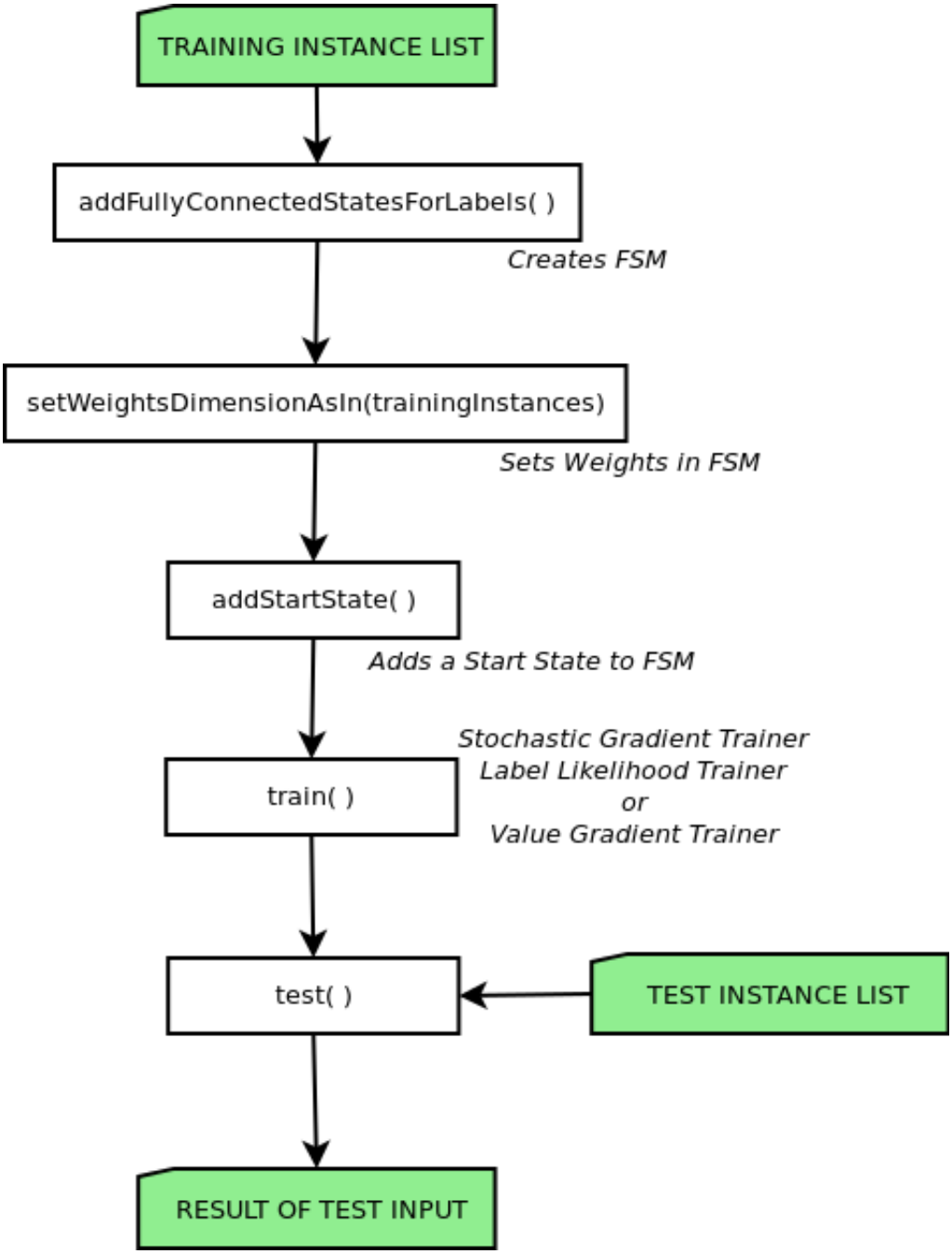

Figure 4.6: Training and Testing Process in MALLET

```
CRFTrainerByStochasticGradient crft = new
     CRFTrainerByStochasticGradient(CRF, LearningRate);
```

To define a stochastic gradient trainer, a CRF object (it is defined previously) and a learning rate is needed. Learning rate determines how fast or slow the system will move towards the optimal weights. If the learning rate is too large, the optimal solu-

tion can be missed out; however, if it is too small, then too many iterations to converge the to the optimal solution are needed. Therefore, it is very important to use an optimal learning rate. It is can be given by the user or this responsibility can be given to the MALLET. In this study, the trainer definition is the following:

```
CRFTrainerByStochasticGradient crft =
     new CRFTrainerByStochasticGradient(CRF, TrainFile);
```

With this definition, MALLET calculates the learning rate which will be the best for this training data set instead of giving it manually.

After creating the CRF Trainer and setting all necessary properties, the trainer can train and test. After CRF finishing its job, it gives the tagged sequences.

Secondly, the system is experimented with a value gradient trainer. Constructing of finite state machine, calculating the weights and adding a start state processes are the same with the case in which the system uses a stochastic gradient trainer.

Since the value gradient trainer method can combine more than one objective function, the objective functions which are not used in other experiments are employed in value gradient experiment, namely batch label-likelihood, and entropy regularization.

An optimizable is defined for entropy regularisation and the other is defined for batch label-likelihood. Since batch label-likelihood is the threaded version of label-likelihood, a threaded optimizable should be defined. The definitions are shown below:

```
CRFOptimizableByBatchLabelLikelihood batchLabel =
        new CRFOptimizableByBatchLabelLikelihood
               (crf, trainingInstances, 1);

CRFOptimizableByEntropyRegularization entropyLabel =
        new CRFOptimizableByEntropyRegularization
               (crf, trainingInstances);
```

```
ThreadedOptimizable optLabelBatch = new
    ThreadedOptimizable(batchLabel, trainingInstances,
         crf.getParameters().getNumFactors(),
              new CRFCacheStaleIndicator(crf));
```

Since the dataset that is used in the experiments is not a large data set enough to parallelized, the number of batch is given 1. This means that **batchLabel** is a label-likelihood optimizable. After defining the optimizables, a CRFTrainerByValueGradients object is defined by combining those two optimizables and then the CRF is trained with it. The trainer definition:

```
Optimizable.ByGradientValue[] opts =
         new Optimizable.ByGradientValue[]
                    {optLabelBatch, entropyLabel};

CRFTrainerByValueGradients crfTrainer =
         new CRFTrainerByValueGradients(crf, opts);
```

The third experimented trainer is label-likelihood trainer. The same finite state machine construction process is also executed in the label-likehood trainer implementation. After construction FSM, a label-likehood trainer is defined and CRF is trained with this trainer:

```
CRFTrainerByLabelLikelihood trainer =
         new CRFTrainerByLabelLikelihood(crf);
```

Label likelihood trainer has an option to use unsupported tricks which aims to add a few weights for features that were not observed in the training data intelligently. For example, in [37], 3.8 million binary features are used; however, the training data used in the proposed system does not include many of these features. Label-likelihood trainer chooses a subset of those kind of features and use them as unsupported features. While training with unsupported tricks option, firstly the CRF model is trained with a few iteration (so that the model is not fully trained) and then the trainer add

unsupported features to all cliques. In this stage, the model does not have the correct answer yet for all cliques. After adding unsupported features, the trainer continues to train with the specified features in the training data [40].

Before training the CRF, this option is set such that trainer will use unsupported trick.

```
trainer.setUseSomeUnsupportedTrick(true);
```

CHAPTER 5

EXPERIMENTS

5.1 Introducing the Datasets

In the experiments, we used two data sets. The first one is from [15]. It contains 2320 tweets and 282 toponyms. We used this data set for conducting experiments which are for finding the best features, training&test data set size, and trainer combination. This dataset is divided into two data sets to use as training and test sets.

The second data is from [29]. We used a subset that consists of 1027 tweets and 446 toponyms. We used this data set for analyzing the effect of using much more number of toponyms for both training and testing. This data set is also divided into two data sets in order to create training and test sets.

The first dataset [15] is obtain from Twitter in a short time period. Therefore, it does not have wide variety of toponyms in it. However, the second dataset is obtained from twitter in a long time period and contains toponyms from different topics. It is taken from [29] which is a event detection study using clustering algorithms. Since it is prepared for using in event detection study, it contains tweets which refer to different events hence the second data set has a wide variety of toponyms.

5.2 Introducing the Experiments

In this section, several experiments are conducted. Firstly, defined features are grouped into some sets in order to analyze their effects on the accuracy of the system, and the

aim is to find the feature combination which gives the higher f-measure value. After finding the best feature combination, training mechanisms which are presented by MALLET are experimented to show their effects on the system results. Finally, to investigate the effects of training data set size and the number of toponym which is used for training, different sized training and test datasets are created and experimented. For all these experiments, first dataset [15] is used. After finding best system combination, it is also experimented with the second dataset [29]. A subset of this dataset is annotated and then it is divided into two as the first one. Finally, the proposed system is compared with two existing NER system.

In experiment results, F-Measure is used to analyze the accuracy of the system. F-Measure is a measure of the accuracy of a test and calculated by the harmonic means of the precision and recall metrics:

$$F\_measure = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.1}$$

Precision represents the proportion of the number of correctly recognized entities over all recognized entities; whereas, recall represents the proportion of the recognized entities over total number of toponyms in the test data set.

5.2.1 Feature Selection Experiments

In this set of experiments, in order to find the best set of features for accuracy, we analyze the performance of different feature sets. In Table 5.1, we list these feature sets and their definitions.

POS-tagging and token character suffix features can be used interchangeably. There are several studies in the literature that aim to find POS-tags of some unknown words that is, probably, belongs to an unknown language. [42] and [41] are the examples for these studies. Moreover, some POS-taggers use the affix features to be able to generate POS-Tags. Therefore, POS-Tags generated by TRmorph and suffixes generated with Mallet pipe are compared in terms of their accuracy. In FeatureSet_1, only POS-tags are used as features for location extraction and in FeatureSet_2 suffix and

Table 5.1: Definitions of Feature Sets

Feature Set	Definitions
FeatureSet_1	POS-Tags
FeatureSet_2	Suffix (suffixes from last 1, last 2, last 3) + Text Character N-Grams (n-values = 3, 7)
FeatureSet_3	Suffix (suffixes from last 1, last 2, last 3) + Text Character N-Grams (n-values = 3, 4, 5)
FeatureSet_4	Suffix (suffixes from last 1, last 2, last 3, last 4, last 5) + Text Character N-Grams (Best n-values from FeatureSet_2 and FeatureSet_3)
FeatureSet_5	POS-Tags + BestResultFrom(FeatureSet_2, FeatureSet_3, FeatureSet_4)
FeatureSet_6	BestResultFrom(FeatureSet_1, FeatureSet_2, FeatureSet_3, FeatureSet_4, FeatureSet_5) + Conjunction (-1, +1)
FeatureSet_7	BestResultFrom(FeatureSet_1, FeatureSet_2, FeatureSet_3, FeatureSet_4, FeatureSet_5) + Conjunction (-2, -1, +1, +2)
FeatureSet_8	BestResultFrom(FeatureSet_6, FeatureSet_7) + RegexMatches
FeatureSet_9	FeatureSet_8 + FirstToken

n-grams features are experimented. However, the results changes according to the suffix size and n values used in n-grams. Therefore, three different feature sets are prepared to analyze the effects of these parameters. FeatureSet_2 uses last one, last two, and last three suffixes and it uses 3, 7 for the n-values. In the third set n-values are changed from 3, 7 to 3, 4, 5. In the FeatureSet_4, n-values which gives the best result is used with suffixes for 1, 2, 3, 4, 5 values.

FeatureSet_5 is the collocation of FeatureSet_1 and the best suffix and n-grams combination. Since, TRmorph can not give the POS-tags for some informal tokens (they cannot be corrected in the Normalization phase), in the FeatureSet_5, POS-tags are used with suffix and text character n-grams features to generate some morphological

features for the tokens which do not have POS-tags.

After comparing the first five feature sets, the feature combination which will have the best result among them is chosen and FeatureSet_6 and FeatureSet_7 are composed by adding Conjunction features to that combination. In sixth and seventh feature sets, the effect of conjunction window size on the accuracy of the system is experimented. In FeatureSet_6, conjunction window is (-1, +1) and the system takes features of the previous and the next tokens into consideration while analyzing a token. In FeatureSet_7, the conjunction window is set to (-2, -1, +1, +2) and considers also the second previous and the second next tokens. FeatureSet_8 is the expanded version of the feature set giving the best result by now with addition of the features obtained from the regular expressions. Lastly, FeatureSet_9 contains all of the features generated in this study.

5.2.2 Experimental Evaluation on CRF Learners

Mallet uses stochastic gradient trainer with CRF while feature sets are experimenting. After choosing the feature set giving the best result, value gradient trainer and label likelihood trainer, are experimented with the same feature set. Finally, a training mechanism which is more successful than the others is chosen to be used for following experiments.

5.2.3 Experimental Evaluation on the size of Training and Test Data Sets

This set of experiments are conducted to determine the best size for training and test data sets. The general idea before the experiments was that the more token sequences the training data set has, the higher accuracy is obtained. With Experiment_1, Experiment_2 and Experiment_3, it is investigated how the number of the toponyms in the training and the test data sets effects the system output. Dataset contains 2320 tweets; in other words, there are 2320 token sequences in the dataset. Training and test data sets are created in different sizes. While creating the data sets, the number of toponyms in those sets are important. To be able to analyze the results of feature sets and trainers accurately, both training and test data set should contain reasonable

number of toponyms. The number of tweets in each training and test data set for each experiment and the number of toponyms in each data set are as follows:

- **Experiment_1:** Training Data Set contains 1670 tweets (71.99%) and Test Data Set contains 650 tweets (28.01%) The number of toponyms in the training data set is 192; in test data set it is 90.
- **Experiment_2:** Training Data Set contains 1160 tweets (50.0%) and Test Data Set contains 1160 tweets (50.0%) The number of toponyms in the training data set is 144; in test data set it is 138.
- **Experiment_3:** Training Data Set contains 800 tweets (34.5%) and Test Data Set contains 1520 tweets (65.5%) The number of toponyms in the training data set is 103; in test data set it is 179.

5.2.4 Experimental Analysis for Accuracy

Lastly, the proposed system is experimented with the second dataset which includes less number of tweets than the first dataset; however, it has much more toponyms. In other words, the toponym density is quite high in the second dataset. By conducting this experiment, best system configuration is used as in previous experiments.

5.2.5 Experimental Comparison with Related Work

After conducting all experiments, the output of the combination giving the best result is compared with the results of the some previous studies. Firstly, the proposed system is compared with Turkish NLP Pipeline [3] and secondly, it is also compared with the system given in [16].

5.3 Experiments and Results

In this section, the results of the experiments described in Section 5.2 are given. Firstly feature sets are going to be experimented. For each experiment, F-Measure,

Precision, Recall and Accuracy metrics are calculated, and the number of True Positives, True Negatives, False Positives and False Negatives are also given in a table for each experiment. The results are evaluated using CoNLL type calculations.

5.3.1 Feature Selection Experiments

In this types of experiments which analyze changes of features, a stochastic gradient trainer is used and all metrics obtained from each feature set experiment are given in table 5.13. The training and testing data set in Experiment_1 is chosen for those experiments.

The first experiment uses only POS-Tags to extract toponyms, and it is run twice to be able to show the impact of the normalization phase on the informal data. Table 5.2 shows the exact numbers of True/False Positive/Negative entities of the each run.

Table 5.2: FeatureSet_1 Results

	TP	**TN**	**FP**	**FN**
FeatureSet_1 (Normalization)	31	6037	5	59
FeatureSet_1 (No Normalization)	23	6037	5	67

When there is no normalisation phase, TRmorph could not find any POS-tag for some tokens which is written informally in Morphological Analysis and Disambiguation phase. The second experiment is conducted to measure the effect of Normalization phase. The accuracy results of this phase are given in Table 5.3.

Table 5.3: Metrics Obtained For FeatureSet_1 With No Normalization

	Precision	**Recall**	**F-Measure**	**Accuracy**
FeatureSet_1	0.821428	0.255555	0.389830	0.988250

Normalization phase makes some adjustments on the initial data. Therefore, some tokens which are toponyms or which are not toponyms but effects the feature vector of the token sequence which is important for sequence labelling are recuperated in the normalization phase and these process increased the number of recognized toponyms.

As an example, the token, "Istanbul" in the following sentence has no POS-Tag since it is written with **"I"** instead of **"İ"**. Therefore, it is not found as a toponym in the second run of the experiment whereas the token "İstanbul" has a B-LOC tag in the normalized version. The given results for both is as follows:

For the experiment with no normalization:

I'm O
at O
Efor O
Kuafor O
Istanbul O

For the experiment with normalization:

I'm O
at O
Efor O
Kuaför O
İstanbul B-LOC

Apparently, there are still location names which are not recognized such as "Efor Kuaför" even if the data is normalized. Therefore, other features are defined to increase the numbers of recognized toponyms.

The second set of features, FeatureSet_2, includes both suffixes and text character n-grams as features for tokens. The purpose is to be able to compare the accuracy of POS-tags and the accuracy of suffix and character n-grams since these features can be used interchangeably. FeatureSet_2 experiment is also run twice, with normalization and with no normalization options. The results are shown in table 5.4.

Table 5.4: FeatureSet_2 Results

	TP	**TN**	**FP**	**FN**
FeatureSet_2 (Normalization)	33	6037	5	57
FeatureSet_2 (No Normalization)	33	6037	5	57

Metrics obtaining from the second run are shown in the table 5.5.

Table 5.5: Metrics Obtained For FeatureSet_2 With No Normalization

	Precision	**Recall**	**F-Measure**	**Accuracy**
FeatureSet_2	0.868421	0.366666	0.515625	0.989889

First and second run of the CRF for suffix and character n-grams features gives the same result. Since there is not a rule to make a token meaningful to extract those kind of features, all tokens have already their features in both case. The difference between the features for an informal token is shown below for the token "viransehir" which is "Viranşehir" in the formal written:

Features of the normalized case:

```
TTCNG_anşehir TTCNG_ranşehi TTCNG_iranşeh TTCNG_viranşe
TTCNG_hir TTCNG_ehi TTCNG_şeh TTCNG_nşe TTCNG_anş
TTCNG_ran TTCNG_ira TTCNG_vir
C3=hir C2=ir C1=r
```

Features of the case with no normalization:

```
TTCNG_ansehir TTCNG_ransehi TTCNG_iranseh TTCNG_viranse
TTCNG_hir TTCNG_ehi TTCNG_seh TTCNG_nse TTCNG_ans
TTCNG_ran TTCNG_ira TTCNG_vir
C3=hir C2=ir C1=r
```

Even though, there is a difference between these feature vectors on letter basis, it did not effect the system accuracy.

The third feature set uses the features in the FeatureSet_2 but with the different n values for character n-grams. In this experiment n values are changed from 3, 7 (in FeatureSet_2) to 3, 4, 5. The number of True Positive, True Negative, False Positive and False Negative findings are shown in table 5.6

Table 5.6: FeatureSet_3 Results

	TP	**TN**	**FP**	**FN**
FeatureSet_3	34	6037	5	56

According to the results in the Table 5.4 and Table 5.6, the system can recognize more toponyms with 3, 4, 5 n-values for character n-grams. Therefore, following experiments uses these n-grams values.

After analyzing the effects of n values and selecting the one finding the higher number of named entities, the impact of the suffix size is investigated. In the previous experiments, suffix size is limited by last three character. If it is expended to the last five character, Table 5.7 is obtained.

Table 5.7: FeatureSet_4 Results

	TP	**TN**	**FP**	**FN**
FeatureSet_4	31	6037	5	59

Since FeatureSet_3 and FeatureSet_4 differs from each other in terms of the suffix size. According to the results of both experiments FeatureSet_3 has 52.71% f-measure whereas f-measure of FeatureSet_4 is 49.2%. Therefore, suffix size is selected as the last 3 three character of a token for the future experiments.

The next experiment combines both POS-Tag features and suffix&character n-grams features. The hypothesis before the experiment is that using the suffixes and the character n-grams to specify the morphological features of some token for which TRmorph do not give any POS-tag should recognizes more toponyms than experiments using only one on them. In this way, it is planned to create morphological feature for almost all tokens with the help of suffix&character n-grams features to the TRmorph. FeatureSet_5 is experimented only for normalized data; since, the first and the second experiments obviously show the impact of the informal to formal conversion. The result of the experiment is in the table 5.8

After verifying the hypothesis with the fifth experiment, other features defined in the

Table 5.8: FeatureSet_5 Results

	TP	**TN**	**FP**	**FN**
FeatureSet_5	38	6037	5	52

system is used by adding them to the FeatureSet_5. The next set combines the fifth set and the conjunction features. Since the conjunction window size is configurable, not only FeatureSet_6 but also FeatureSet_7 will use the conjunction features; however, the size of the windows changes. In FeatureSet_6, a window of [-1, +1] is used while in FeatureSet_7 the size is [-2, +2] which includes the features of the second previous, previous, next and the second next tokens. The purpose is to obtain higher accuracy on the recognition of named entities which consists of more than one token; since, the conjunction feature takes the features of the tokens inside the window into consideration.

Results of the experiments on sixth and seventh feature sets are shown in Table 5.9 and 5.10 respectively.

Table 5.9: FeatureSet_6 Results

	TP	**TN**	**FP**	**FN**
FeatureSet_6	39	6036	6	51

Table 5.10: FeatureSet_7 Results

	TP	**TN**	**FP**	**FN**
FeatureSet_7	39	6032	10	51

Optimum window size, actually, depends on the number of named entities which includes more than one entity and the number of tokens exist in a named entity. Since the dataset, used in those experiments, contains a few named entity which consists of more than three token. Therefore, expanding the window is not increase the number of recognized entities. However, some unrelated features such as the POS-Tag of the two previous token are added to a token even though it is not related with the

current token. Since, this case happens too much, the system finds some big named entities which is not a named entity indeed. Therefore, the number of the false positive findings increases.

The base metric to measure the success of the system is F-Scores and FeatureSet_6 has higher F-Score since the metric takes the false positive findings into account besides the true positives. Therefore, following experiments uses the FeatureSet_6 as a basis.

The next experiment uses FeatureSet_8 which adds regex features to FeatureSet_6 since it is the most successful feature set for toponym recognition among first seven sets. It is hypothesized that, labelling some words or part of words which generate toponyms with the token when they come after it or when they included in it makes the probability of the recognition of those entities higher. In this feature set, tokens which obeys the regular expression rules are tagged as "LOCATION". Results of the experiment is shown in table 5.11.

Table 5.11: FeatureSet_8 Results

	TP	**TN**	**FP**	**FN**
FeatureSet_8	42	6032	10	48

After obtaining the experiment results, it is obvious that labelling some generative character sequences gives a gain about 1.4% in F-Measure value, which is a measure of the achievement of the system.

Last experiment on feature sets is conducted with FeatureSet_9 which extends the eighth feature set by adding "FirstToken" feature. This feature tags the first tokens in the each token sequence. How this feature effects the result of the system is shown in table 5.12

According to the results, labelling the first tokens in the data set decreases the number of true positive findings. Therefore, the feature set which gives the best result is FeatureSet_8 and its victory can be monitored from Table 5.13 and from the Figure 5.1.

Table 5.12: FeatureSet_9 Results

	TP	TN	FP	FN
FeatureSet_9	38	6032	10	52

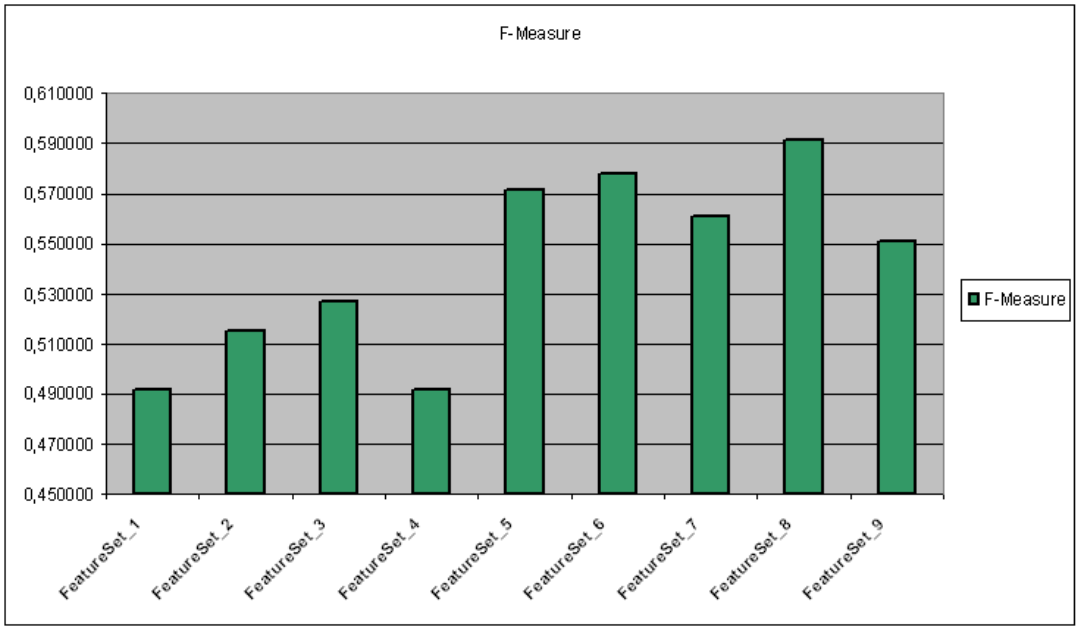

Figure 5.1: Metrics Obtained For Each Feature Set

Table 5.13: F-Measure Metrics Obtained For Each Feature Set

	Precision	Recall	F-Measure	Accuracy
FeatureSet_1	0.861111	0.344444	0.492063	0.989562
FeatureSet_2	0.868421	0.366667	0.515625	0.989889
FeatureSet_3	0.871794	0.377778	0.527131	0.990052
FeatureSet_4	0.868421	0.366667	0.515625	0.989889
FeatureSet_5	0.883721	0.422222	0.571429	0.990704
FeatureSet_6	0.866667	0.433333	0.577778	0.990705
FeatureSet_7	0.795918	0.433333	0.561151	0.990052
FeatureSet_8	0.807692	0.466666	0.591549	0.990541
FeatureSet_9	0.791667	0.422222	0.550724	0.989889

The results of the experiments which are described so far are given in Figure 5.2, Figure 5.3, they shows the changes between experiments in terms of Precision, and Recall respectively. Higher precision and recall metrics are preferable. However, the feature set which gives the best precision may not give the best recall. For example, even though the highest precision is obtained by FeatureSet_5 (Figure 5.2), its recall is lower than FeatureSet_8. F-Measure comes with a solution for this problem by taking the harmonic mean of those metrics. Therefore, it is the only criteria to decide that which experiment gives the best result. The answer is FeatureSet_8 according to the Figure 5.1.

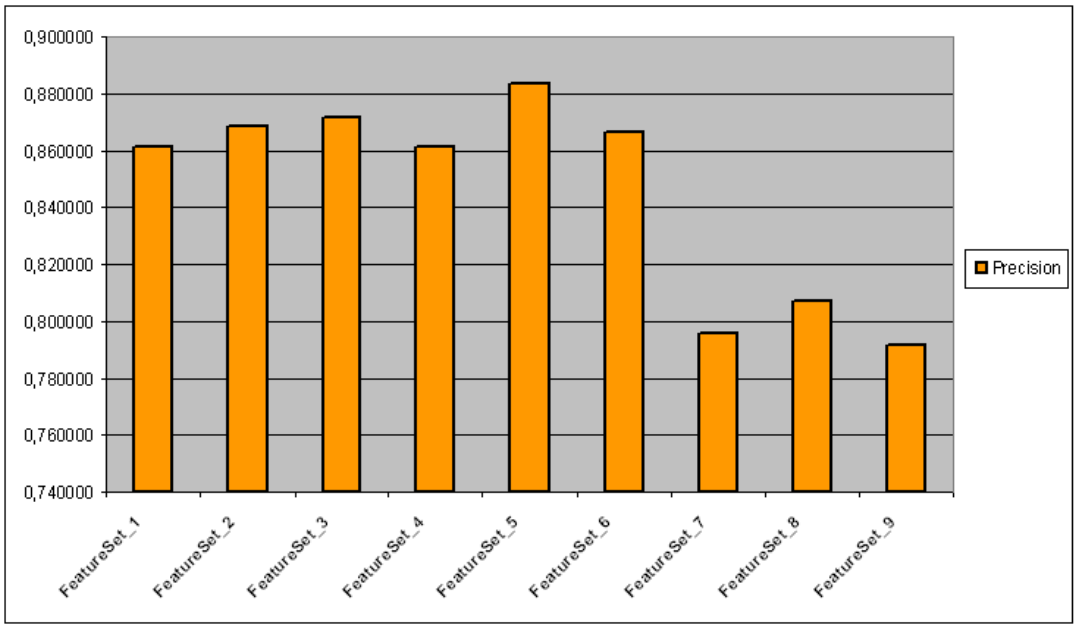

Figure 5.2: Precision Metrics Obtained For Each Feature Set

FeatureSet_8 will be used in future experiments which is conducted to investigate how the size of the input data sets and different kind of trainers affect the system output.

Before concluding the feature selection experiments, another statistics needs to be given for normalization phase. The dataset used in the experiments has an informal structure. Therefore, the Morphological Analyse phase can not produce results for some tokens in the dataset. After these tokens are given to the Normalization phase; following statistics are obtained.

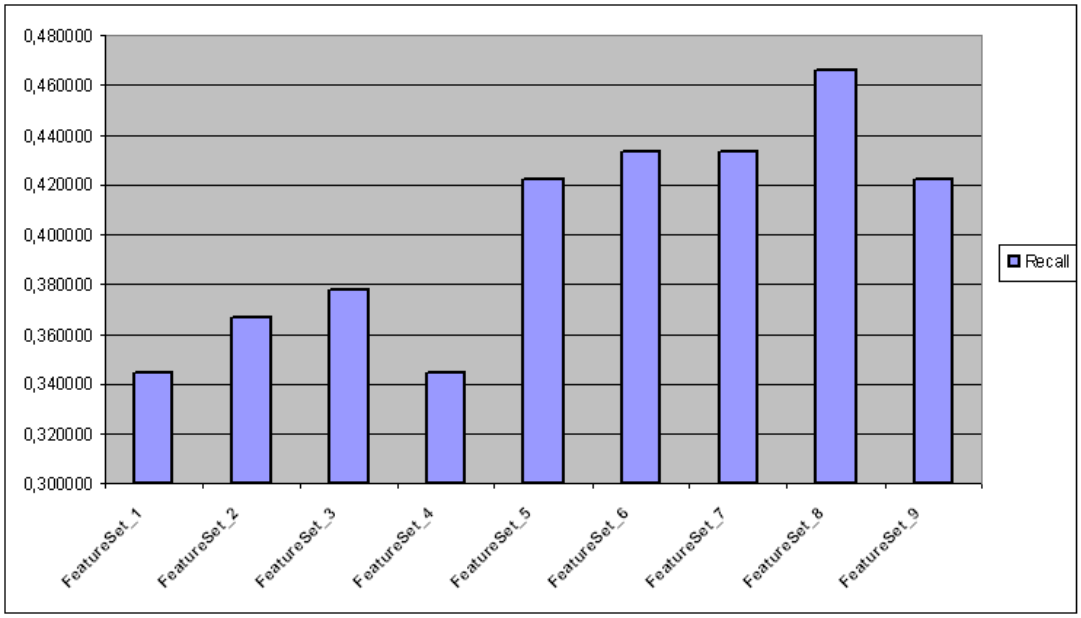

Figure 5.3: Recall Metrics Obtained For Each Feature Set

The number of tokens that dataset contains : **20.380**
The number of tokens that the Morphological Analyse does not give results before the normalization: **2886**
The number of tokens that the Morphological Analyse give results after Normalization (number of the adjusted tokens):**1351**

In conclusion, the Normalization phase works with a success percent of 47% on this dataset.

5.3.2 Experimental Evaluation on CRF Learners

There are three types of trainer used in the proposed work, and the one giving the best results will be the trainer of the following experiments. In the previous tests, stochastic gradient trainer is employed and it gives the best result with the f-measure of 59.15%. Other trainers, namely label likelihood and value gradient, are experimented respectively with the FeatureSet_8. The detailed information about those three trainer is given at Section 2.4. The trainers give the results which are shown in Table 5.14.

Table 5.14: F-Measures from Each Trainer Experiments with the FeatureSet_8 (which gives the best result)

	Precision	**Recall**	**F-Measure**	**Accuracy**
Stochastic Gradient Trainer	0.807692	0.466667	0.591549	0.990541
Label Likelihood Trainer	0.933333	0.466667	0.622222	0.991683
Value Gradient Trainer	0.95	0.422222	0.584615	0.991193

According to Table 5.14 label-likelihood trainer has an average precision whereas value gradient is more successful with respect to the precision metric; however, it gives the lowest recall value. Stochastic gradient trainer has the highest recall whereas its precision is quite low. Label likelihood trainer has the best f-measure; since it has an average precision and the highest recall. Therefore, in the next experiments, a label-likelihood trainer will be employed.

5.3.3 Experimental Evaluation on the size of Training and Test Data Sets

The last parameter to construct a successful system is to arrange the size of the training and test data sets. The number of the toponyms in those data sets should also be meaningful. Since training data set is used to train CRF, the more comprehensive training data set yields a better learning. Therefore, location extraction process from test data set gives a more accurate results. To show the mentioned effect of training data set size three experiment will be conducted. However, in addition to the size of the training data set, the number of toponyms in that set is also critical. The more various type of location entities CRF learns, the more variety of toponyms the system can extract. In the experiment preparation phase, this condition is taken into account. Therefore, different types of input data sets combinations are generated.

Firstly, the dataset divided into two sets in a way that the training data set includes 1670 tweets and 192 location names inside those tweets and the test data set contains 650 tweets and 90 location named entities. While the dataset is separating, a shuffling method is used and it is controlled that both training and test data set have reasonable number of toponyms.

All previous experiments is conducted by using these firstly created files and the result is already taken. For the second experiment, the input data sets are created in a way that both the training and test data sets have the same number of tweets. However, the training data set has a few extra toponym than the test data set. In this experiment both the training and test data sets consist of 1160 tweets; however, the training data set includes 144 toponyms; whereas, the test set contains 138 location entities.

The last experiment uses a training data set which is much smaller than the test set. The numbers of tweets are 800 and 1520 for the training and test data sets respectively. 179 location named entites are in the test set and 103 entities are included in the training set. Obtained results from those three experiments can be seen in Table 5.15.

Table 5.15: F-Measures from Each Experiment with the Different Numbers of Tweets and Toponyms

	Precision	**Recall**	**F-Measure**	**Accuracy**
Experiment_1 (192/90)	0.933333	0.466667	0.622222	0.991683
Experiment_2 (144/138)	0.921569	0.340579	0.497354	0.991460
Experiment_3 (103/179)	0.934782	0.240223	0.382222	0.990620

The experiments give expected results. When the number of location named entities is decreased in training data set, CRF is trained with not enough information about toponyms. Therefore, its ability to recognize the named entities is also rasped.

In order to show the results of the best system configuration more clearly, the list of toponyms in the test data set and true positive, false positive and false negative findings are given in Appendix A. According to the list, toponyms consisting of more than 3 entity cannot be found since the window size is 3. Some tokens referring to a location cannot be found; whereas, the others referring to the same location can be labelled as toponym since the sequences containing them are quite different and the training data set is not comprehensive enough to train CRF with all those sequences.

5.3.4 Experimental Analysis for Accuracy

Previous experiments are conducted by using the first dataset [15]. This dataset is larger than the second one; however, it contains less number of toponyms (282). In

the second one, even though, there is less number of tweets, it has more number of location named entities. Another experiment is designed for the second dataset by using the best combination. Findings and metrics obtained from this experiment are in Table 5.16 and Table 5.17 respectively. Total toponym count in the test data set is 130 while the training set is containing 316.

Table 5.16: Second Dataset Results

	TP	**TN**	**FP**	**FN**
Second Dataset	93	3409	8	37

Table 5.17: Second Dataset Metrics

	Precision	**Recall**	**F-Measure**	**Accuracy**
Second Dataset	0.920792	0.715385	0.805195	0.987313

The results are evidence for that the more toponyms the machine learning model is trained with, the higher accuracy a NER system gives. The second dataset makes the model train better since it has more location names. Moreover, training and test data sets composed from the second dataset contain similar toponyms since the probability of being tweets which belong to the same event in both training and test data sets is quite high.

5.3.5 Experimental Comparison with Related Work

The proposed system is compared with Turkish NLP Pipeline ([3], [7]). It has similar steps such as tokenization, normalization, morphological processing, etc. with the proposed work in general; however, it uses different kind of features and methods to operate these steps. The results of the proposed system is given in the previous section. The test data set used in those experiment which is the one including 650 tweets and 90 toponyms is also experimented in Turkish NLP Pipeline. Firstly, the test data set is given to the normalization tool and after taking the results from normalizer, it is processed with the morphological analyzer. The analyzer gives all possible morphological features for each token. Then, morphological disambiguator is run with the

output of the analyzer. Lastly, the final results are taken from Named Entity Recognizer. All mentioned tools can be found at their web site [7]. The comparison between the proposed work and Turkish NLP Pipeline is given in Table 5.19 and Table 5.18 in terms of precision,recall and f-measure metrics and the numbers of true/false positive and true/false negative findings respectively.

Table 5.18: Comparison between Turkish NLP Pipeline and The Proposed System in terms of The Numbers of True Positive, True Negative, False Positive and False Negative Findings

	TP	**TN**	**FP**	**FN**
Proposed Work	42	6039	3	48
Turkish NLP Pipeline	23	6030	12	67

Table 5.19: Comparison between Turkish NLP Pipeline and The Proposed System in terms of Precision, Recall, F-Mesaure and Accuracy

	Precision	**Recall**	**F-Measure**	**Accuracy**
Proposed Work	0.933333	0.466667	0.622222	0.991683
Turkish NLP Pipeline	0.657142	0.255556	0.367999	0.987116

According to the result tables, the proposed work gives a higher accuracy. The factors which yields these results are investigated. After investigation, following issues are found in Turkish NLP pipeline:

- Tokenization is not capable to tokenize some inputs correctly. For example:

  The original tweet: "I'm at Kapalıçarşı (İstanbul, Türkiye) w/ 86 others http://t.co/VpCuN9hPCj"

  Its tokenized form after normalization in pipeline with the generated POS-Tags:

```
I'm I'm+?
at at+Noun+A3sg+Pnon+Nom
Kapalıçarşı Kapalıçarşı+Noun+Prop+A3sg+Pnon+Nom
```

```
( (+Punc
İstanbul İstanbul+Noun+Prop+A3sg+Pnon+Nom
, ,+Punc
Türkiye) Türkiye)+?
v/ v/+?
86 86+Num+Card
otokar otokar+Noun+A3sg+Pnon+Nom
@url[http://t.co/VpCuN9hPCj]
              @url[http://t.co/VpCuN9hPCj]+?
```

Its tokenized form in the proposed system with the generated POS-Tags:

```
I’m Alpha p1s
at N
Kapalıçarşı N:prop
İstanbul N:prop
Türkiye N:prop
w Alpha
86 Num:ara
others ???
http://t.co/VpCuN9hPCj ???
. Punc
```

Even though, the token "Türkiye" is a token which is written correctly, the morphological analyzer does not give any POS-Tag result since it is meaningless in this form, **"Türkiye)"**, because of the parenthesis at the end. This is not the unique case in the output of the normalization and tokenization phase. Since this token is not recognized, it does not tagged as a location in the final output (Named Entity Recognizer output).

- In normalization phase, there are some adjustments on tokens resulting with words which are not correct. For example:

  The original tweet: "Catili bi evim olsun cati katinin tamamni kendi odam yapiyim ilk kati kocaman mutfak ikinci katida salon odalar falan oha super be"

  NER output for this tweet:

```
Çatılı çatı+Noun+A3sg+Pnon+Nom^DB+Adj+With O
bir bir+Num+Card O
evim ev+Noun+A3sg+P1sg+Nom O
olsun ol+Verb+Pos+Imp+A3sg O
Cat'i Cad+Guess+Noun+Prop+A3sg+Pnon+Acc O
katinin kati+Noun+A3sg+Pnon+Gen O
tamamen tamamen+Adverb O
kendi kendi+Pron+Reflex+A3sg+P3sg+Nom O
odam oda+Noun+A3sg+P1sg+Nom O
yapıyım yapı+Noun+A3sg+Pnon+Nom^DB+
                            Verb+Zero+Pres+A1sg O
ilk ilk+Adj O
kati kati+Adj O
kocaman kocaman+Adj O
mutfak mutfak+Noun+A3sg+Pnon+Nom O
ikinci ikinci+Num+Ord O
Kati'da Kati'da+? B-LOCATION
salon salon+Noun+A3sg+Pnon+Nom O
odalar oda+Noun+A3pl+Pnon+Nom O
falan falan+Adj O
oha oha+Interj O
Super Super+Guess+Noun+A3sg+Pnon+Nom O
be be+Interj O
```

Actually, the normalization phase in the pipeline makes successful adjustments. For example, it is capable to convert the informal sentence, "ankaraya gitcem", to the formal sentence, "Ankara'ya gideceğim". However, the normalization process yields a lot of wrong conversion also. For instance, "cati katinin" is converted to "Cat'i katinin". Especially the conversion of the token, "katida", makes the pipeline tag this token as a location, which is quite inconsequent. The encounter probability of this kind of miscorrection is not low. However, all of them do not yield a mislabelling. In fact, it is easily corrected with English-Turkish character checking task. The proposed system normalizes this tweet as

the following:

```
Çatılı N li Adj O
bi Num:typo O
evim N p1s O
olsun V imp 3s O
çatı N O
katının Adj 0 N gen O
tamamni O
kendi Prn:refl:3s O
odam N p1s O
yapıyım N 0 V cpl:pres 1s O
ilk Adj O
kati Adj O
kocaman Adj O
mutfak N O
ikinci Num ord O
katıda Adj 0 N loc O
salon N O
odalar N pl O
falan N O
oha Ij O
süper Adj O
be Ij O
. Punc O
```

As a result, the pipeline has an issue about tokenization; whereas, the normalization makes amazing adjustment for some tokens or sentences. However, the adjustments yields a wrong result for many tokens in the dataset. Therefore, the negative effects of the normalizing process brings the positive effects of the system into disrepute.

- Different CRFs tools are used in the pipeline and in the proposed system. CRF is implemented by MALLET [25] in the proposed system where as in the

pipeline, CRF++ is employed. It is hard to express the effects of those tools on the accuracy with an example. However, MALLET can give the opportunity to choose different train methods even if the CRF model is still linear-chain. It was shown that the train methods has serious effect on the accuracy of the system with conducted experiments (Table 5.14). The difference between the methods constructing CRF models on both system can be affecting the accuracy of the systems.

The proposed system is also compared with a rule-based NER system ([17], [16]) by using the results of the experiment in which the test data set containing 650 tweets and 90 toponyms. However, the result of the rule-based NER system could not be taken by using the same test data set. Since the whole dataset which is divided into two in this thesis is used in [16] as a test data set in experiments, a result file is generated by choosing the same tweets which are in our test set from the output file of the rule-based system. According to the created test data set, Table 5.20 and Table 5.21 are obtained.

Table 5.20: Comparison between Rule-based NER System and The Proposed System in terms of The Numbers of True Positive, True Negative, False Positive and False Negative Findings

	TP	**TN**	**FP**	**FN**
Proposed Work	42	6039	3	48
Rule-based NER System	59	6025	17	31

Table 5.21: Comparison between Rule-based NER System and The Proposed System in terms of Precision, Recall, F-Mesaure and Accuracy

	Precision	**Recall**	**F-Measure**	**Accuracy**
Proposed Work	0.933333	0.466667	0.622222	0.991683
Rule-based NER System	0.776315	0.655555	0.710843	0.992172

The proposed system has higher precision value; however, its recall value is quite lower than the rule-based system. Since the rule-based system is using some pre-defined lists and pre-defined rules, the obtained result highly depends on lists and the

dataset. Since it is impossible to re-run the rule-based system for now, it cannot be experimented with the second dataset.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, it is aimed to extract named entities referring to location names from an informal and unstructured data by training a learning based NER system. The system needs some features to be able to identify toponyms in the dataset. The better features can describe toponyms and others which are not toponyms, the better the system recognizes location named entities. Therefore, it is critical to define such features. The proposed system aims to find the best feature combination for toponym recognition. Another purpose is to find a best training and testing method which works with CRFs model and to observe effects of the content and the size of the training and test data sets.

The proposed system is a hybrid NER system since it uses several rule-based features while training a learning based model. It is another purpose of the study to analyze the effects of rule-based features on a learning-based system.

Since the system works on informal data, a normalization phase is added to the work-flow. This phase aims to adjust some informally written Turkish tokens to be able to recognize them. It is a vital process to guess the correct forms of the tokens since each token makes a contribution to the probability of finding toponyms even if it does not refer to a location name.

This system uses linear-chain CRFs for machine learning model. However, it is claimed that skip-chain CRFs give more accurate results than linear-chain CRFs [39]. Therefore, the model of the proposed system can be changed from linear-chain model to skip-chain model. Other improvements on the system can be done by giving

some additional features to the CRFs model which makes toponyms more clear or by making the normalization phase more comprehensive. Moreover, by using domain-specific data, since it increases the similarity between training and test data sets, the system accuracy can be also improved. All those modifications are leaved as future work.

The proposed learning based system has near 62.22% F-Measure metric with the first dataset and 80.5% F-Measure value with the second dataset for informal Twitter domain when it uses the best combination of trainer, feature set and training data set size. Considering the results of informal English NER systems such as [8] with F-Measure of 66%, [38] with F-Measure of 75.81%, [34] with F-Measure of 74%, and the results of informal Turkish NER systems such as [3] with F-Measure of 36.8%, [16] with F-Measure of 71%, it can be said that the proposed system gives quite good results. Since the results are dependent on the dataset used in the experiments, it is hard to say that it gives the best result among informal Turkish NER studies. To specify its place in the literature, the dataset must be analyzed with other Turkish NER systems and different kinds of datasets should be experimented with the proposed system. Because of the annotated dataset limitations, the accuracy of the system with a large training data is not tested.

REFERENCES

[1] R. Balabantaray, S. Das, and K. T. Mishra. Case Study of Named Entity Recognition in Odia Using CRF++ Tool. *International Journal of Advanced Computer Science and Applications(IJACSA)*, 4:213–216, 2013.

[2] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[3] G. Çelikkaya, D. Torunoğlu, and G. Eryiğit. Named Entity Recognition on Real Data: A Preliminary Investigation for Turkish. *Proceedings of the 7th International Conference on Application of Information and Communication Technologies*, October 2013.

[4] N. Chinchor. MUC-7 Named Entity Task Definition. URL: `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/ne_task.html` [cited 15.07.2014].

[5] Ç. Çöltekin. A Freely Available Morphological Analyzer for Turkish. *European Language Resources Association*, pages 820–827, 2010.

[6] Ç. Çöltekin. TRmorph: A Turkish Morphological Analyzer, 10 2013. URL: `http://www.let.rug.nl/coltekin/trmorph/` [cited 23.06.2014].

[7] G. Eryiğit. ITU Turkish NLP Web Service. *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, April 2014.

[8] D. Etter, F. Ferraro, and R. Cotterell et al. Nerit: Named Entity Recognition for Informal Text. Technical report, The Johns Hopkins University, the Human Language Technology Center of Excellence, HLTCOE 810Wyman Park Drive Baltimore, Maryland 21211, July 2013.

[9] J. R. Finkel, T. Grenager, and C. Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics*, pages 363–370, 2005.

[10] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named Entity Recognition through Classifier Combination. *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL*, 4:168–171, 2003.

[11] A. Göksel and C. Kerslake. *Turkish: A Comprehensive Grammer*. Comprehensive Grammars. Routhledge (Taylor and Francis), London, 2005.

[12] M. Haspelmath and A.D. Sims. *Understanding Morphology*. Hodder Education, an Hachette UK Company, 338 Euston Road, London NW1 3BH, 2 edition, 2010.

[13] J. Kapociute-Dzikiene, A. Noklestad, J. Bondi Johannessen, and A. Krupavicius. Exploring Features for Named Entity Recognition in Lithuanian Text Corpus. *Proceedings of the 19th Nordic Conference of Computational Linguistics*, pages 73–88, May 2013.

[14] V. Krishnan and C. D. Manning. An Effective Two-stage Model for Exploiting Non-local Dependencies in Named Entity Recognition. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 1121–1128, 2006.

[15] D. Kucuk, G. Jacquet, and R. Steinberger. Named Entity Recognition on Turkish Tweets. *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, may 2014.

[16] D. Küçük and R. Steinberger. Experiments to Improve Named Entity Recognition on Turkish Tweets. *In Proceedings of the EACL Workshop on Language Analysis for Social Media*, 2014.

[17] D. Küçük and A. Yazıcı. Named Entity Recognition Experiments on Turkish Texts. *In Proceedings of the International Conference on Flexible Query Answering Systems*, 5822:524–535, 2009.

[18] S. Kumova Metin, T. Kışla, and B. Karaoğlan. Named Entity Recognition In Turkish Using Association Measures. *Advanced Computing: An International Journal (ACIJ)*, 3(4), July 2012.

[19] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, 2001.

[20] M. Lieberman and H. Samet. Multifaceted Toponym Recognition for Streaming News. *ACM*, pages 843–852, 2011.

[21] M. D. Lieberman, H. Samet, and J. Sankaranarayanan. Geotagging with Local Lexicons to Build Indexes for Textually-specified Spatial Data. *In Proceedings of ICDE*, pages 201–212, 2010.

[22] J. Lingad, S. Karimi, and J. Yin. Location Extraction from Disaster-related Microblogs. *International World Wide Web Conference*, pages 1017–1020, 2013.

[23] A. McCallum. Efficiently Inducing Features of Conditional Random Fields. *Proceeding UAI'03 Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 403–410, 2002.

[24] A. Mccallum, K. Rohanimanesh, and C. Sutton. Dynamic Conditional Random Fields for Jointly Labeling Multiple Sequences. *NIPS Workshop on Syntax, Semantics and Statistics*, 2003.

[25] A. K. McCallum. MALLET: A Machine Learning for Language Toolkit., 2002. URL: `http://mallet.cs.umass.edu` [cited 25.06.2014].

[26] W. Murnane. Improving Accuracy of Named Entity Recognition on Social Media Data, 2010.

[27] K. Oflazer. Two-level Description of Turkish Morphology. *Proceedings of the Sixth Conference on European Chapter of the Association for Computational Linguistics*, page 472, 1993.

[28] K. D. Önal, P. Karagöz, and R. Çakıcı. Türkçe Twitter Gönderilerinde Lokasyon Tanıma. *SIU*, April 2014.

[29] O. Ozdikis, P. Senkul, and H. Oguztüzün. Semantic Expansion of Tweet Contents for Enhanced Event Detection in Twitter. *The IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, pages 20–24, 2012.

[30] S. Özkaya and B. Diri. Named Entity Recognition by Conditional Random Fields From Turkish Informal Texts. *IEEE Signal Processing and Communications Applications Conference (SIU)*, pages 662–665, 2011.

[31] C. Pal, X. Wang, M. Kelm, and A. Mccallum. Multi-Conditional Learning for Joint Probability Models with Latent Variables. *Annual Conference on Neural Information Processing Systems Workshop on Advances in Structured Learning for Text and Speech Processing*, 2006.

[32] D. Pinto, A. McCallum, X. Wei, and W. B. Croft. Table Extraction Using Conditional Random Fields. *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 235–242, 2003.

[33] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled LDA: A Supervised Topic Model for Credit Attribution in Multi-Labeled Corpora. *EMNLP*, 1:248–256, 2009.

[34] A. Ritter, S. Clark, and M. O. Etzioni. Named Entity Recognition in Tweets: An Experimental Study. *in Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

[35] S. Sarawagi. Information Extraction. *Foundation and Trends in Databases*, 1(3):261–377, March 2008.

[36] G. A. Şeker and G. Eryiğit. Initial Explorations on Using CRFs for Turkish Named Entity Recognition. *24th International Conference on Computational Linguistics*, pages 2459–2474, 2012.

[37] F. Sha and F. Pereira. Shallow Parsing with Conditional Random Fields. *NAACL*, 1, 2003.

[38] N.V Sobhana, M. Pabitra, and S.K. Ghosh. Conditional Random Field Based Named Entity Recognition in Geological Text. *International Journal of Computer Applications*, 1(3):119–125, February 2010.

[39] C. Sutton and A. Mccallum. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[40] C. Sutton and A. McCallum. An Introduction to Conditional Random Fields. *Foundations and Trends in Machine Learning*, 4:267–373, August 2012.

[41] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, 1:173–180, 2003.

[42] H. Tseng, D. Jurafsky, and C. Manning. Morphological Features Help POS Tagging of Unknown Words across Language Varieties. *Proceedings of the fourth SIGHAN bakeoff*, 2005.

[43] L. Xiaohua, Z. Shaodian, W. Furu, and Z. Ming. Recognizing Named Entities in Tweets. *Association for Computational Linguistics: Human Language Technologies*, 1:359–367, 2011.

APPENDIX A

FINDINGS IN THE TEST DATA SET

Toponyms in Test Data Set	
eskişehirde	suriye
Kapalıçarşı	EDİRNEDEYİZ
İstanbul	İzmir'i
Türkiye	Şile
Samsun	Değirmen Çayırı köyü
karadenize	Gazelle Gölcük Kır Gazinosu
Mısır	Gölcük
Filistin	New York
İstanbul	Türkiye
Türkiye	ayvalığı
Türkiye	Merter'de
istanbul'un	D-100 karayolu
Ankara	Merter
Türkiye	İncirli
izmir	Adapazarı'nda
ankaradayız	yeni zelanda'nın
Bağlarbaşı Caddesi	Kumlubük Maris beachte
İstanbul	Panora Tepe
Türkiye	ÇUBUK
Kadıköy'de	Gallifrey'e
İzmirde	BURSA
Ankarada	Tunus
İstanbul	Fethiyedeyim

istanbulu	india
İstanbul	fındıkzade
Türkiye	çeşme kampüsü
manavgatta	Efor Kuaför
İstanbulla	Isparta
İzmir	Gelendost
Türkiye	bodrum
Mecidiyeköy	Ortadoğu
İstanbul	türkbeleni
Türkiye	Forum Bornova
Bahçelievler deneme anadolu lisesindeyim	Enez'de
Ankara	Şaygan Anadolu lisesi
Türkiye	EDİRNE
ANKARA	REİNA
türkiye	Hayrabolu
İstanbul	Tomalı Hilmi Caddesi
İstanbul	İNCİRLİYE
Ankaraya	İZMİR
ankaradayız	Malta
Lös Angeles	Power Spor Merkezi
Samsun	Avrupa
kız kulesi	ortadoğuda

True Positives in Test Data Set

eskişehirde	Ankarada
Kapalıçarşı	İstanbul
İstanbul	istanbulu
Türkiye	İstanbul
Değirmen Çayırı köyü	Türkiye
karadenize	manavgatta
Mısır	İstanbulla
Filistin	Şaygan Anadolu lisesi
İstanbul	Türkiye

Türkiye	Mecidiyeköy
Türkiye	İstanbul
istanbul'un	Türkiye
Ankara	Enez'de
Türkiye	Ankara
Tomalı Hilmi Caddesi	Türkiye
ankaradayız	ANKARA
Bağlarbaşı Caddesi	türkiye
İstanbul	İstanbul
Türkiye	İstanbul
Kadıköy'de	Ankaraya
İzmirde	ankaradayız

False Positives in Test Data Set

İstanbul	İstanbul
Radyosu	

False Negatives in Test Data Set

Lös Angeles	Tunus
Samsun	Fethiyedeyim
kız kulesi	india
suriye	fındıkzade
EDİRNEDEYİZ	çeşme kampüsü
İzmir'i	Efor Kuaför
Şile	Isparta
Samsun	Gelendost
Gazelle Gölcük Kır Gazinosu	bodrum
Gölcük	Ortadoğu
New York	türkbeleni
Türkiye	Forum Bornova
Bahçelievler deneme anadolu lisesindeyim	ayvalığı
Merter'de	İzmir
D-100 karayolu	EDİRNE
Merter	REİNA

İncirli	Hayrabolu
Adapazarı’nda	izmir
yeni zelanda’nın	İNCİRLİYE
Kumlubük Maris beachte	İZMİR
Panora Tepe	Malta
ÇUBUK	Power Spor Merkezi
Gallifrey’e	Avrupa
BURSA	ortadoğuda